AD A119623

# NAVAL POSTGRADUATE SCHOOL
## Monterey, California

DTIC
ELECTE
SEP 27 1982
S D
D

# THESIS

A SENSITIVITY ANALYSIS OF
THE KALMAN FILTER AS APPLIED
TO AN INERTIAL NAVIGATION SYSTEM

Gary Glen Potter

June, 1982

Thesis Advisor: D. J. Collins

Approved for public release; distribution unlimited



82 09 27 043

# REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER

2. GOVT ACCESSION NO. AD-A119623

3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle)

A Sensitivity Analysis of the Kalman Filt
as Applied to an Inertial Navigation Sys

5. TYPE OF REPORT & PERIOD COVERED

Master's Thesis
June, 1982

6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s)

Gary Glen Potter

8. CONTRACT OR GRANT NUMBER(s)

9. PERFORMING ORGANIZATION NAME AND ADDRESS

Naval Postgraduate School
Monterey, California 93940

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME AND ADDRESS

Naval Postgraduate School
Monterey, California 93940

12. REPORT DATE

June, 1982

13. NUMBER OF PAGES

104

14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)

15. SECURITY CLASS. (of this report)

Unclassified

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Inertial Navigation Systems
Kalman Filter
Sensitivity Analysis
Covariance Analysis

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A tactical missile with mid-course requires the use of an Inertial Navigation System (INS). Steady-state Kalman Filters (SKF) used as estimators have been proposed for use in a Strapdown INS that is considered to be cheaper and easier to implement than a gimbaled INS.

This thesis further investigates the sensitivity of the SKF to inaccuracies in the filter parameters such as the dimensional stability

20. ABSTRACT (Continued)

derivatives. The analysis is expanded to explore the sensitivity of a system of higher dimension created by the augmentation of an additional state. The study has been performed by independently varying each of the filter parameters over a given range and noting the effect on the accuracy of the filter. One of the benefits of this analysis of the rms estimate errors to variations in the stability derivatives is that it reveals which derivatives need to be accurately determined to ensure stable flight.

Approved for public release; distribution unlimited

A Sensitivity Analysis of
the Kalman Filter as Applied
to an Inertial Navigation System

by

Gary Glen Potter
Lieutenant Commander, United States Navy
B.S.E.E., Univeristy of Idaho, 1973

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ENGINEERING SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
June 1982

Author: Gary G. Potter

Approved by: Daniel J. Collins, Thesis Advisor

J. A. Titus, Second Reader

Robert D. Strum
Chairman, Department ~~of Electrical~~ Engineering

William M. Tolles
Dean of Science and Engineering

ABSTRACT

A tactical missile with mid-course requires the use of an Inertial Navigation System (INS). Steady-state Kalman Filters (SKF) used as estimators have been proposed for use in a Strapdown INS that is considered to be cheaper and easier to implement than a gimbaled INS.

This thesis further investigates the sensitivity of the SKF to inaccuracies in the filter parameters such as the dimensional stability derivatives. The analysis is expanded to explore the sensitivity of a system of higher dimension created by the augmentation of an additional state. The study has been performed by independently varying each of the filter parameters over a given range and noting the effect on the accuracy of the filter. One of the benefits of this analysis of the rms estimate errors to variations in the stability derivatives is that it reveals which derivatives need to be accurately determined to ensure stable flight.

## TABLE OF CONTENTS

LIST OF FIGURES

## LIST OF TABLES

## ACKNOWLEDGEMENT

I would like to express my sincere appreciation to Dr. D. J. Collins for his patient guidance and assistance, and steady encouragement during the conduct of this work. Through his leadership our Lord enabled me to complete this thesis and accept an essentially new career as an Engineering Duty Officer.

# I. INTRODUCTION

A tactical missile normally requires midcourse guidance to ensure that its trajectory leads to a specific target. A typical midcourse guidance law pre-programmed strategy maintains constant altitude, heading and speed. Such guidance is primarily effected by an Intertial Navigation System (INS). In this work two steady-state Kalman Filters (SKF), used as estimators of the longitudinal and lateral motion, constitute what may be considered as part of a Strapdown INS onboard a missile that can be cheaper and easier to implement than a gimballed INS. The authors of [Ref. 1] discuss the basic differences between Strapdown and gimballed Inertial Navigation Systems.

Sensors on the missile that the longitudinal and lateral estimators could use are described by Maybeck [Ref. 2] and include laser rate gyros, doppler velocimeters, magnetic compasses, and barometric altimeters. A radar seeker could provide a distance or range measurement or range rate. Distance or position measurement could be computed from a signal inserted into the missile's INS from the Global Positioning System (GPS) or similar satellite-based navigation system.

This work was motivated by Bryson [Ref. 3], where he discusses a Strapdown INS using SKF as estimators applied to the model for the DC-8 airplane. To avoid classification requirements and for convenience, the model used here is essentially the same as that of [Ref. 3] rather than that of a missile.

This thesis is a continuation of the work done by Matallana [Ref. 4]. It further investigates the sensitivity of the Kalman Filter to inaccuracies in the filter parameters or varation between the filter model and the plant model for longitudinal motion estimation. The differences could be due to model inaccuracies or to normal variation caused by a changing flight environment. The sensitivity of rms estimate errors to inaccuracies or differences in the stability derivatives is the result of interest.

The initial work conducted was to reproduce the results of [Ref. 3] and [Ref. 4] with the correct implementation of the dynamics in the filter parameters. Then the results of [Ref. 4] for the longitudinal motion estimator with incorrect implementation of the dynamics in the Kalman Filter were reproduced.

After a distance measurement and associated system and measurement noise parameters were added to the model dynamics, the sensitivity analysis was repeated for the longitudinal motion estimator. The analysis of the effect that this distance input had on the sensitivity of the rms errors to inaccuracies or differences in the stability derivatives of the Kalman Filter concluded the research for this thesis.

## II. MODELS AND ESTMATION

### A. KALMAN FILTER

Only a brief description of the Kalman Filter has been included to show the particular formulation used. A more complete development of general theory is done by Gelb in [Ref. 5].

#### 1. Linear Dynamic System

Consider the linear time invariant system (plant and measurement models) given by equation (1) below, where x represents the states of the system; z is the measurement; F is the system matrix; $\Gamma$ is the driving noise coefficient matrix; H is the measurement scaling matrix; and w and v are independent, zero-mean, white gaussian noise processes with covariance matrices Q and R respectively.

$$\dot{x} = Fx + \Gamma w \tag{1-a}$$

$$z = Hx + v \tag{1-b}$$

Mathematically, Q and R are represented by equation (2) as:

$$E(w(t)w^T(\tau)) = Q(t)\sigma(t-\tau), \; E(w(t)) = 0 \tag{2-a}$$

$$E(v(t)v^T(\tau)) = R(t)\sigma(t-\tau), \; E(v(t)) = 0 \tag{2-b}$$

#### 2. Continuous Kalman Filter

A continuous time Kalman Filter is described by equation (3) where $\hat{x}$ is the state estimate and K is a matrix of constant filter gains.

$$\dot{\hat{x}} = F\hat{x} + K(z-H\hat{x}) \tag{3}$$

The implementation of the System Model and the Kalman Filter is shown in Figure 1.

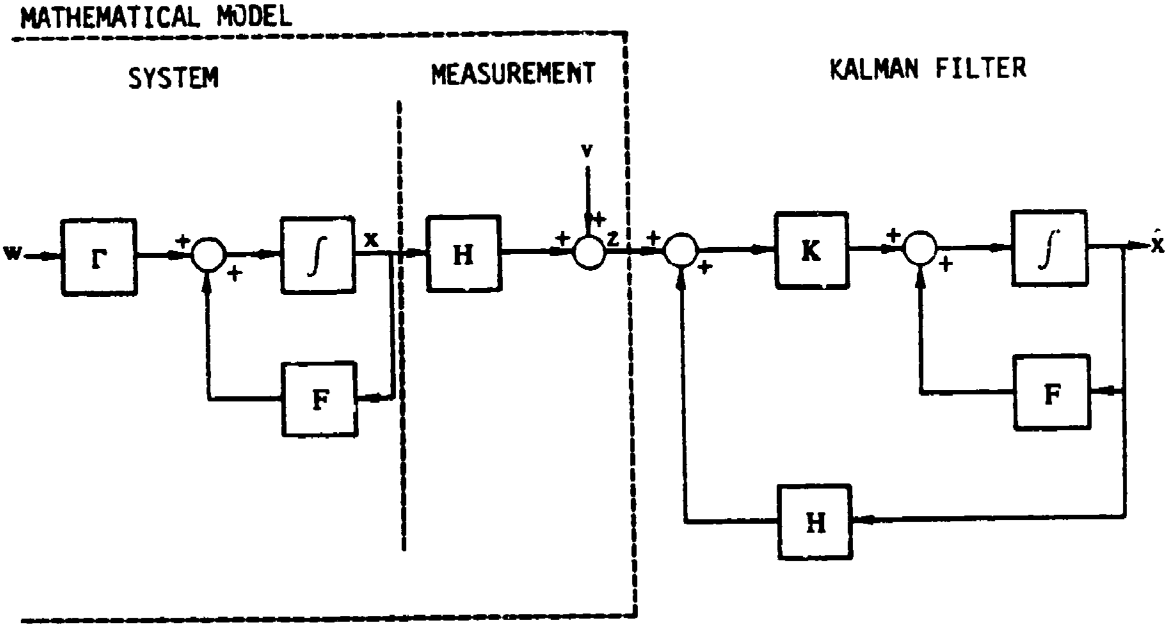


Figure 1. System Model and Kalman Filter

The estimate error is defined by equation (4) as

$$\tilde{x} \triangleq \hat{x} - x \tag{4}$$

and the differential equation for $\tilde{x}$ is given by

$$\dot{\tilde{x}} = (F-KH)\tilde{x} - \Gamma w + Kv \tag{5}$$

The differential equations for the states of a linear system driven by noise can be expressed as

$$\begin{bmatrix} \dot{\tilde{x}} \\ \dot{x} \end{bmatrix} = \begin{bmatrix} F-KH & 0 \\ 0 & F \end{bmatrix} \begin{bmatrix} \tilde{x} \\ x \end{bmatrix} + \begin{bmatrix} Kv-\Gamma w \\ \Gamma w \end{bmatrix} \tag{6}$$

The covariance of the estimate-error, symbolized as P, is defined by equation (7). It provides a statistical measure of the uncertainty in x.

$$P = E(\tilde{x}\tilde{x}^T) \tag{7}$$

The diagonal elements of the covariance matrix are the root mean square errors of the state variables. Also, the trace of P is the mean square length of the vector $\tilde{x}$. The off diagonal terms of P indicate the degree of cross-correlation between the elements of $\tilde{x}$. The covariance matrix P is obtained by solving the linear Lyapunov equation given by

$$P = (F-KH)\ P + P(F-KH)^T + \Gamma Q \Gamma^T + KRK^T \tag{8}$$

The eigenvalues of the filter are given by the roots of

$$|SI - F + KH| = 0 \tag{9}$$

## B. STATE AUGMENTATION AND SHAPING FILTERS

When the system random disturbances are correlated in time, i.e., colored noise, it is necessary to use their power spectral density data in order to develop a mathematical model that produces an output which duplicates the noise characteristics [Ref. 2]. Correlated random noises are taken to be state variables of a ficticious linear time invariant system (usually called a shaping filter) which is itself excited by white gaussian noise. Such a model is given by equation (10) below, where the

subscript f denotes filter, and n is a nonwhite (time-correlated) gaussian noise. The filter output is used to drive the system depicted by Figure 2.

$$\dot{x}_f = F_f x_f + \Gamma_f w \tag{10-a}$$

$$z_n = H_f x_f \tag{10-b}$$

The dimension of the state vector (1) is increased by including the disturbances as well as a description of the system dynamics behavior in appropriate rows of an enlarged F matrix. This enlargement process is called state vector augmentation.

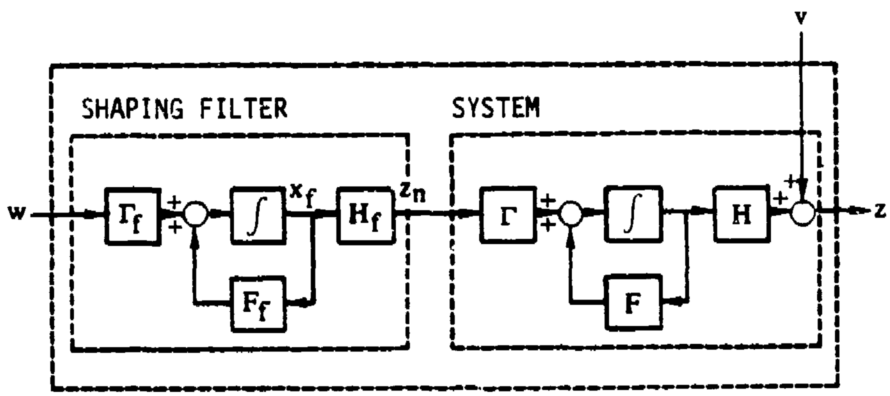


Figure 2. Shaping Filter Generating Driving Noise

The augmented state equation is given by

$$\left[\frac{\dot{x}}{\dot{x}_f}\right] = \left[\begin{array}{c|c} F & \Gamma H_f \\ \hline & F_f \end{array}\right]\left[\frac{x}{x_f}\right] + \left[\frac{0}{\Gamma_f}\right] w \tag{11}$$

The associated measurement equation is

$$z = \left[ H \middle| 0 \right] \left[ \frac{x}{x_f} \right] + v \qquad (12)$$

C. SENSITIVITY TO PARAMETER VARIATION

Observing the structure of the Kalman Filter illustrated in Figure 1, the filter contains an exact model of the system dynamics.

The analysis of how the error covariance behaves when the gain matrix is computed using perturbed values of the F matrix, such as varying parameters due to different flight conditions, is well explained in [Ref. 5]. Figure 3 is a block diagram of the system model and Kalman Filter with the system dynamics perturbed. F* is the perturbed system dynamics, while K* is the associated gain matrix computed for the Kalman Filter.

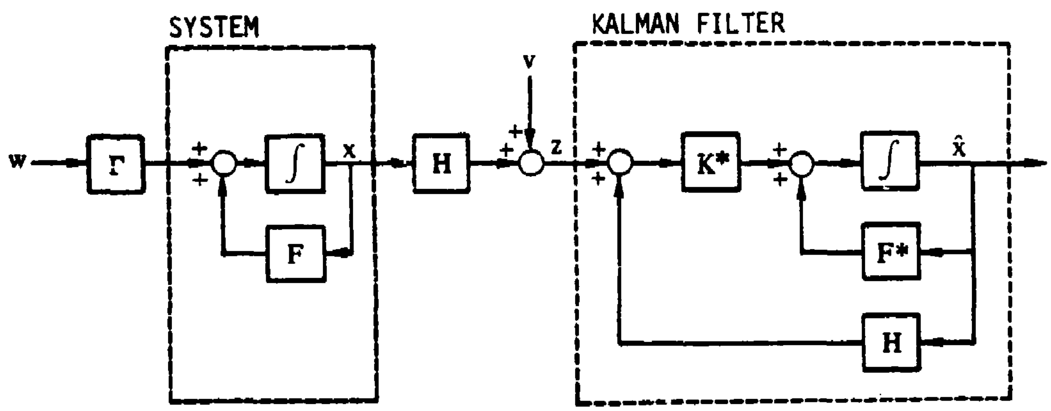


Figure 3. System Model and Kalman Filter with Perturbed Dynamics

The equation for the estimate is given by

$$\dot{\hat{x}} = F^*\hat{x} + K^*(z-H\hat{x}) \tag{13}$$

The error in the estimate is given by

$$\dot{\tilde{x}} = (F^* - K^*H)\tilde{x} + \Delta Fx - \Gamma w + K^*v \tag{14}$$

where

$$\Delta F \triangleq F^* - F \tag{15}$$

The differential equations for the states of linear system driven by white gaussian noise now become

$$\begin{bmatrix} \dot{\tilde{x}} \\ \hline \dot{x} \end{bmatrix} = \begin{bmatrix} F^* - K^*H & \Delta F \\ \hline 0 & F \end{bmatrix} \begin{bmatrix} \tilde{x} \\ \hline x \end{bmatrix} + \begin{bmatrix} K^*v-\Gamma w \\ \hline \Gamma w \end{bmatrix} \tag{16}$$

Letting x' be the augmented state vector, $x' \triangleq \begin{bmatrix} \tilde{x} \\ \hline x \end{bmatrix}$.

The covariance matrix of x' is given by

$$E(x'x'^T) = \left[\begin{array}{c|c} P & V^T \\ \hline V & U \end{array}\right] \tag{17}$$

where one defines $P \triangleq E(\tilde{x}\tilde{x}^T)$, $V \triangleq E(x\tilde{x}^T)$, and $U \triangleq E(xx^T)$. P, the covariance of x, is the quantity of interest. The error sensitivity equations are:

$$\dot{P} = (F^* - K^*H)\ P + P(F^* - K^*H)^T + \Delta FV + V^T\Delta F + \Gamma Q\Gamma^T + K^*RK^{*T} \tag{18-a}$$

$$\dot{V} = FV + V(F^* - K^*H)^T + U\Delta F^T - \Gamma Q\Gamma^T \tag{18-b}$$

and

$$\dot{U} = FU + UF^T + \Gamma Q\Gamma^T \tag{18-c}$$

with initial conditions $P(0) = -V(0) = U(0) = E(x(0)x(0)^T)$. When the actual system dynamics are reproduced in the filter, $F = F^*$ and $\Delta F = 0$, and equation (18) reduces to the linear Lyapunov equation of equation (8).

## D. MODAL COORDINATES TRANSFORMATION

The system represented by equation (1) is not unique. Consider an alternate linear transformaton of the states described in references [3] and [6]. Let $x = T$, where $\xi$ represents the transformation of the states and T is the transformation matrix with the columns formed by the eigenvectors of the system matrix F (for a complex eigenvalue, the first column is the real part and the second is the imaginary part of the eigenvector). The similarity transformation of equation (1) is

$$\dot{\xi} = A\xi + Bw \tag{19-a}$$

$$z = C\xi + v \tag{19-b}$$

where $A = T^{-1} FT$, $B = T^{-1}\Gamma$, and $C = HT$.

A case of particular interest, the canonical form, results when the A matrix is diagonal (i.e., when the eigenvalues of the F matrix appear on the diagonal). This canonical form is more informative than the transfer function method, since observability and controlability of the system can be obtained by inspection.

## E. SOLUTION OF THE SKF WITH A PRESCRIBED DEGREE OF STABILITY

The constant gain Kalman Filter (SKF) used as an observor will diverge if undisturbed, neutrally stable (UNS) modes are in the system model. In references [3] and [7] the authors discussed the destabilization of the system model (1). The amount of destabilization can be varied until the suboptimal observor formed has a desired degree of stability. The method of [Ref. 3] destabilizes only the UNS modes in the system model and is called "modal destabilization" (MDS). In this technique the gains of the filter are constrained so that

$$Re(S_i) > -\sigma, \quad i = 1,2,\ldots\ldots\ldots,n \tag{20}$$

where $Re(S_i)$ indicates the "real" part of $(S_i)$, $S_1,\ldots,S_n$ are the eigenvalues of the filter, i.e., the roots of equation (9), and $\sigma$ is a specified positive number.

The original system model is destabilized in accordance with equation (21), where $F'$ is the destabilized matrix formed, $E$ is the destabilization matrix (diagonal), and $T$ is the modal transformation matrix (eigenvector matrix). The matrix $F'$ is used to calculate the suboptimal gains of the filter.

$$F' = F + TET^{-1} \tag{21}$$

This MDS approach prevents the divergence of the steady-state Kalman Filter in a system with UNS model while causing only a slight reduction in the estimation accuracy.

## III. DYNAMIC AND MEASUREMENT SYSTEM MODELS

### A. REFERENCE AXIS SYSTEM

The Reference Axis System of a missile is centered at its center of gravity (c.g.) and fixed on the missile body as follows:

X axis, the roll axis, forward from the c.g. along the axis of symmetry.

Y axis, the pitch axis, outward to the right from the c.g. when viewing the missile from behind.

Z axis, the yaw axis, downward from the c.g. in the plane of symmetry to form a right-handed orthogonal system with the other two.

Appendix A lists the symbols defining quantities associated with the missile illustrated in Figure 4 below such as forces and moments, linear and angular velocities, and moments of inertia.

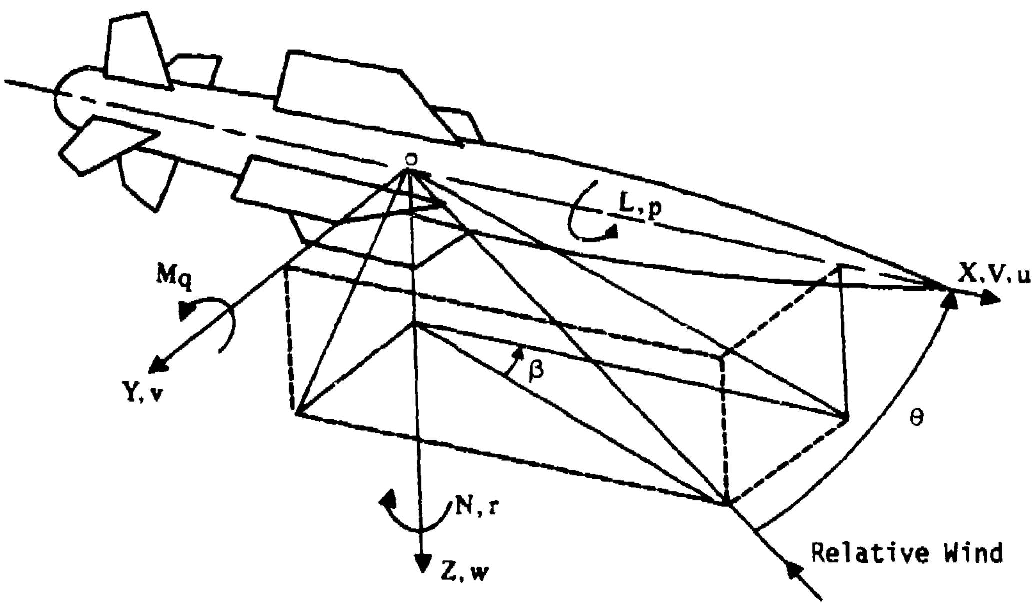


Figure 4. Reference Axis System

## B. MISSILE EQUATIONS OF MOTION

The equations of motion used to represent the missile dynamics used in this study are well defined in [Ref. 8]. A linear dynamical model of the missile based on the rigid body approximation is appropriate.

### 1. Longitudinal Motion

The longitudinal motions of a missile can be modeled by a fifth-order system of equation (22), where the state variables are u, velocity along the X axis, w, velocity along the Z axis, q, pitch rate, $\theta$, pitch angle and h, altitude. The units are: u and w in 10 ft/s, q in 0.01 rad/s, $\theta$ in 0.01 rad, and h in 100 ft.

$$\begin{bmatrix} \dot{u} \\ \dot{w} \\ \dot{q} \\ \dot{\theta} \\ \dot{h} \end{bmatrix} = \begin{bmatrix} Xu & Xw & 0 & -g & 0 \\ Zu & Zw & V & 0 & 0 \\ Mu+M\dot{w}Zu & Mw+M\dot{w}Zw & Mq+MwV & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & -0.1 & 0 & V & 0 \end{bmatrix} \begin{bmatrix} u \\ w \\ q \\ \theta \\ h \end{bmatrix} \quad (22)$$

### 2. Lateral Motion

The lateral motions of a missile are modeled by the fifth-order system given by equation (23), where the state variables are: $\beta$, sideslip angle, r, yaw rate, p, roll rate, $\phi$, roll angle, and $\psi$, heading angle. The units are: $\beta$ in rad, r in rad/s, p in rad/s, $\theta$ in rad, and $\psi$ in rad.

$$\begin{bmatrix}\dot{\beta}\\ \dot{r}\\ \dot{p}\\ \dot{\phi}\\ \dot{\psi}\end{bmatrix} = \begin{bmatrix} Yv & -1 & 0 & g/V & 0\\ N^1_\beta & N^1_r & N^1_p & 0 & 0\\ L^1_\beta & L^1_r & L^1_p & 0 & 0\\ 0 & 0 & 1 & 0 & 0\\ 0 & 1 & 0 & 0 & 0\end{bmatrix}\begin{bmatrix}\beta\\ r\\ p\\ \phi\\ \psi\end{bmatrix} \tag{23}$$

C. MODEL DYNAMICS

The aerodynamic data used in this paper appears in Appendix B. Except for the addition of system and measurement noise parameters for the distance input, the models and noise dynamics are the same as those of [Ref. 3].

1. Longitudinal Motion Estimation

The main disturbance inputs are the two wind velocities $u_g$ and $w_g$. Under certain flight conditions, the turbulance represented by the fluctuating parts of $u_g$ and $w_g$ are colored noise. They are modeled by first-order shaping filters with white gaussian noise inputs as shown in equation (10). The linear model that results is given by equation (24) [Ref. 4].

$$\begin{bmatrix}\dot{u}_g\\ \dot{w}_g\end{bmatrix} = \begin{bmatrix}-0.413 & 0\\ 0 & -0.853\end{bmatrix}\begin{bmatrix}u_g\\ w_g\end{bmatrix} + \begin{bmatrix}0.413 & 0\\ 0 & 0.853\end{bmatrix}\begin{bmatrix}\mu_u\\ \mu_w\end{bmatrix} \tag{24}$$

The numerical data for the longitudinal dimensional derivatives was used in equation (22). The resultant model is represented by equation (25) which corresponds to the state vector augmentation of equation (11). Scaling is done with u, w, $u_g$, and $w_g$ in units of 10 ft/s, q in units of 0.01 rad/s, θ in units of 0.01 rad, and h in units of 100 ft.

$$\begin{bmatrix} \dot{u} \\ \dot{w} \\ \dot{q} \\ \dot{\theta} \\ \dot{h} \\ \dot{u}_g \\ \dot{w}_g \end{bmatrix} = \begin{bmatrix} -0.015 & 0.004 & 0 & -0.0322 & 0 & -0.015 & 0.004 \\ -0.074 & -0.806 & 0.824 & 0 & 0 & -0.074 & -0.806 \\ -0.749 & -10.7 & -1.344 & 0 & 0 & -0.749 & -10.7 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -0.1 & 0 & 0.0824 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -0.413 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -0.853 \end{bmatrix} \begin{bmatrix} u \\ w \\ q \\ \theta \\ h \\ u_g \\ w_g \end{bmatrix}$$

$$+ \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0.413 & 0 \\ 0 & 0.853 \end{bmatrix} \begin{bmatrix} \mu_u \\ \mu_w \end{bmatrix} \qquad (25)$$

The measurement model shown by equation (26) assumes a rate gyro in order to measure $z_q$ and a barometric altimeter to measure $z_h$.

$$\begin{bmatrix} z_q \\ z_h \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} u \\ w \\ q \\ \theta \\ h \\ u_g \\ w_g \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_q \\ v_h \end{bmatrix} \tag{26}$$

2. Lateral Motion Estimation

The main disturbance input is the lateral wind v. The turbulence represented by the fluctuating part of v is the colored noise, which is also modeled as a first-order shaping filter with white gaussian noise input as given by equation (10). The resulting shaping filter taken from [Ref. 3] is given by equation (27).

$$\dot{\beta}_g = -0.853\beta_g + 0.853\mu \tag{27}$$

where $\beta_g = v_g/V$.

The numerical data for the lateral dimensional derivatives was applied in equation (23) to obtain equation (28), which corresponds to the state vector augmentation of equation (11).

$$\begin{bmatrix} \dot{\beta} \\ \dot{r} \\ \dot{p} \\ \dot{\phi} \\ \dot{\psi} \\ \dot{\beta}_g \end{bmatrix} = \begin{bmatrix} -0.0868 & -1 & 0 & 0.03907 & 0 & -0.0868 \\ 2.14 & -0.228 & -0.0204 & 0 & 0 & 2.14 \\ -4.41 & 0.334 & -1.181 & 0 & 0 & -4.41 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -0.853 \end{bmatrix} \begin{bmatrix} \beta \\ r \\ p \\ \phi \\ \psi \\ \beta_g \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0.853 \end{bmatrix} \mu \tag{28}$$

The measurement model given by equation (29) below represents the case where the measurement z is taken with a roll-rate gyro and the measurement z obtained from a magnetic compass.

$$\begin{bmatrix} z_p \\ z_\psi \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \beta \\ r \\ p \\ \phi \\ \psi \\ \beta_g \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_p \\ v_\psi \end{bmatrix} \qquad (29)$$

# IV. ANALYSIS

## A. SIMULATION

The Sensitivity Covariance Program developed for application in the work of [Ref. 4] was used to solve the error sensitivity equations of equation (18). The program was originally developed to handle a set of 105 linear differential equations for the longitudinal case and 78 for the lateral. The program was revised to accommodate 136 linear differential equations in the longitudinal case and 101 in the lateral, to allow for an additional state augmentation (i.e., the distance measurement to the longitudinal model). The outputs of these programs are the time matrices and rms estimate errors, the square roots of the diagonal elements of the P matrices. The OPTSYS program, the use of which is described by [Ref. 9] and amplified by [Ref. 10], was applied to calculate the Kalman Filter gains to be inserted into the Sensitivity Covariance Program to find the estimate errors for specific system parameter perturbations. The OPTSYS program was also used to destabilize the systems that contained UNS modes in an attempt to eliminate filter divergence. Copies of the OPTSYS and the Sensitivity Covariance programs follow under COMPUTER PROGRAMS, while a copy of [Ref. 10] appears in Appendix C.

## B. RESULTS

As the problem is introduced, the results are presented in three parts: (1) to verify the findings of [Ref. 3] and [Ref. 4] with the correct implementation of the dynamics in the filter parameters, (2) to

reproduce the findings of [Ref. 4] for the longitudinal motion estimator with incorrect implementation of the dynamics in the Kalman Filter, and (3) to conduct a sensitivity analysis of the longitudinal motion estimator after adding a distance measurement and associated system and measurement noise parameters to the model dynamics.

### 1. Motion Estimation Analysis for Exact Dynamics

The OPTSYS program was used with input data representing the actual system dynamics for both the longitudinal and lateral cases to obtain the following results which are the same as those of [Ref. 4] and essentially the same as those of [Ref. 3].

#### a. Longitudinal Case

| filter gain matrix K | | filter eigenvalues |
|---|---|---|
| 0.059 | 0.060 | -0.310 + j0.411 |
| 0.264 | -0.161 | -0.429 |
| 3.517 | 0.040 | -0.178 |
| 0.001 | -0.080 | -0.261 |
| -0.011 | 0.035 | -0.063 + j0.0743 |
| -1.288 | 0.128 | |

rms estimate errors

| | |
|---|---|
| $\bar{u}$ = 2.090 ft/s | $\bar{\theta}$ = 0.317 deg |
| $\bar{w}$ = 5.102 ft/s | $\bar{h}$ = 8.245 ft |
| $\bar{q}$ = 0.416 deg/s | $\bar{u}_g$ = 4.776 ft/s |
| $\bar{w}_g$ = 5.701 ft/s | |

b. Lateral Case

| filter gain matrix K | | filter eigenvalues |
|---|---|---|
| 0.051 | -0.967 | $-2.350 + j2.594$ |
| -1.536 | 0.411 | $-0.624 + j0.492$ |
| 2.695 | -0.004 | -0.00125 |
| 0.386 | -0.789 | 0.0 |
| -0.005 | 0.906 | |
| -1.713 | 0.655 | |

rms estimate errors

$\bar{v}_\beta$ = 3.329 ft/s

$\bar{r}$ = 0.244 deg/s

$\bar{p}$ = 0.377 deg/s

$\bar{\phi}$ = 0.222 deg

$\bar{\psi}$ = 0.214 deg

$\bar{V}_{\beta_g}$ = 5.506 ft/s

2. Longitudinal Motion Estimation Analysis

The OPTSYS program was used to compute a new K* matrix as each parameter of the F matrix was individually numerically varied. The Sensitivity Covariance Program was then executed utilizing each new K* and F* matrix pair to determine the rms errors for each individual perturbation.

The results are shown in Tables 1-8 and are identical to those of [Ref. 4]. The true values for the unperturbed system dynamics parameters are indicated in the tables by an asterisk. A discussion of the results follows:

$\underline{X_u}$. The dimensional variation of the X force with forward speed u has a nominal value of -0.015. This quantity was varied in a range of ±20%. The behavior of the rms estimate errors can be seen in Table 1. The tabulation shows that the numerical variation of the $X_u$ derivative does not cause significant changes in the nominal values of the rms estimate errors of the states $\bar{w}$, $\bar{q}$, $\bar{\theta}$, $\bar{u}_g$, and $\bar{w}_g$. The states $\bar{u}$ and $\bar{h}$ appear to be slightly effected, but not enough to be of importance.

$\underline{X_w}$. The dimensional variation of the X force with downward speed w has a nominal value of 0.004. Again, a numerical variation in a range of ±20% was conducted. The behavior of the rms errors is demonstrated by Table 2. Comparing these values with the nominal ones reveals that changes in the $X_w$ derivative have essentially no effect on the states $\bar{w}$, $\bar{q}$, $\bar{\theta}$, $\bar{u}_g$, and $\bar{w}_g$, while the states $\bar{u}$ and $\bar{h}$ show changes too small to consider important.

$\underline{Z_u}$. The dimensional variation of the Z force caused by a change in the forward speed u has a nominal value of -0.074. The design value was altered in a range of ±20% with the results shown in Table 3. Evaluation of this data indicates that all the rms errors show some sensitivity except for that of $\bar{q}$. The most significant changes occur in the $\bar{u}$, $\bar{\theta}$, and $\bar{h}$ states. The large variation in $\bar{u}$ can be important in terms of the accuracy in radial position.

$\underline{Z_w}$. The dimensional variation of the Z force with downward speed w has a nominal value of -0.806. The results for this case with changes in $Z_w$ over a range of ±20% follow in Table 4. They show that all the rms estimate errors are quite sensitive and any variation of $Z_w$ beyond ±2% can be considered critical and unacceptable.

TABLE 1. RMS ESTIMATE ERRORS FOR LONGITUDINAL MOTION ESTIMATOR WITH VARIATION IN $X_u$ DERIVATIVE

| $X_u$ | $\bar{u}$ ft/s | $\bar{w}$ ft/s | $\bar{q}$ deg/s | $\bar{\theta}$ deg | $\bar{h}$ ft | $\bar{u}_g$ ft/s | $\bar{w}_g$ ft/s |
|---|---|---|---|---|---|---|---|
| -0.018 | 2.096 | 5.102 | 0.416 | 0.317 | 8.248 | 4.776 | 5.701 |
| -0.0165 | 2.094 | 5.102 | 0.416 | 0.317 | 8.246 | 4.776 | 5.701 |
| -0.01575 | 2.091 | 5.102 | 0.416 | 0.317 | 8.240 | 4.776 | 5.701 |
| -0.015 * | 2.090 | 5.102 | 0.416 | 0.317 | 8.245 | 4.776 | 5.701 |
| -0.01425 | 2.088 | 5.103 | 0.416 | 0.317 | 8.260 | 4.775 | 5.701 |
| -0.0135 | 2.089 | 5.103 | 0.416 | 0.317 | 8.280 | 4.775 | 5.701 |
| -0.012 | 2.092 | 5.103 | 0.416 | 0.317 | 8.340 | 4.775 | 5.701 |

TABLE 2. RMS ESTIMATE ERRORS FOR LONGITUDINAL MOTION ESTIMATOR WITH VARIATION IN $X_w$ DERIVATIVE

| $X_w$ | $\bar{u}$ ft/s | $\bar{w}$ ft/s | $\bar{q}$ deg/s | $\bar{\theta}$ deg | $\bar{h}$ ft | $\bar{u}_g$ ft/s | $\bar{w}_g$ ft/s |
|---|---|---|---|---|---|---|---|
| 0.0048 | 2.070 | 5.102 | 0.416 | 0.317 | 8.319 | 4.776 | 5.701 |
| 0.0044 | 2.080 | 5.102 | 0.416 | 0.317 | 8.282 | 4.776 | 5.701 |
| 0.0042 | 2.086 | 5.102 | 0.416 | 0.317 | 8.257 | 4.776 | 5.701 |
| 0.004 * | 2.090 | 5.102 | 0.416 | 0.317 | 8.245 | 4.776 | 5.701 |
| 0.0038 | 2.100 | 5.102 | 0.416 | 0.317 | 8.223 | 4.776 | 5.701 |
| 0.0036 | 2.103 | 5.102 | 0.416 | 0.316 | 8.215 | 4.776 | 5.701 |
| 0.0032 | 2.110 | 5.101 | 0.416 | 0.316 | 8.180 | 4.776 | 5.701 |

TABLE 3. RMS ESTIMATE ERRORS FOR LONGITUDINAL MOTION ESTIMATOR WITH VARIATION IN $Z_u$ DERIVATIVE

| $Z_u$ | $\bar{u}$ ft/s | $\bar{w}$ ft/s | $\bar{q}$ deg/s | $\bar{\theta}$ deg | $\bar{h}$ ft | $\bar{u}_g$ ft/s | $\bar{w}_g$ ft/s |
|---|---|---|---|---|---|---|---|
| -0.0888 | 1.885 | 5.106 | 0.416 | 0.322 | 9.310 | 4.776 | 5.707 |
| -0.0814 | 1.974 | 5.104 | 0.416 | 0.319 | 8.808 | 4.775 | 5.703 |
| -0.0777 | 2.026 | 5.194 | 0.416 | 0.318 | 8.579 | 4.775 | 5.702 |
| -0.740 * | 2.090 | 5.102 | 0.416 | 0.317 | 8.245 | 4.776 | 5.701 |
| -0.0703 | 2.122 | 5.100 | 0.416 | 0.315 | 7.800 | 4.777 | 5.700 |
| -0.0666 | 2.270 | 5.095 | 0.416 | 0.313 | 7.101 | 4.777 | 5.697 |
| -0.0592 | 2.32 | 5.094 | 0.416 | 0.311 | 7.000 | 4.778 | 5.695 |

TABLE 4. RMS ESTIMATE ERRORS FOR LONGITUDINAL MOTION ESTIMATOR WITH VARIATION IN $Z_w$ DERIVATIVE

| $Z_w$ | $\bar{u}$ ft/s | $\bar{w}$ ft/s | $\bar{q}$ deg/s | $\bar{\theta}$ deg | $\bar{h}$ ft | $\bar{u}_g$ ft/s | $\bar{w}_g$ ft/s |
|---|---|---|---|---|---|---|---|
| -0.9612 | 30.08 | 6.172 | 0.486 | 0.440 | 17.97 | 4.778 | 7.138 |
| -0.8866 | 11.80 | 5.810 | 0.428 | 0.415 | 33.50 | 4.785 | 5.957 |
| -0.8463 | 5.345 | 5.259 | 0.421 | 0.400 | 22.776 | 4.779 | 5.737 |
| -0.806 * | 2.090 | 5.102 | 0.416 | 0.317 | 8.245 | 4.776 | 5.701 |
| -0.7657 | 2.668 | 5.032 | 0.412 | 0.219 | 16.903 | 4.772 | 5.710 |
| -0.7256 | 3.188 | 5.035 | 0.407 | 0.200 | 21.026 | 4.760 | 5.746 |
| -0.665 | 3.260 | 5.065 | 0.406 | 0.185 | 23.000 | 4.767 | 5.794 |

$\underline{M_u}$. The dimensional variation of the M moment caused by a change in the forward speed u has a nominal value of -0.000786. From Table 5, one notes that the rms errors for the states $\bar{q}$, $\bar{u}_g$, and $\bar{w}_g$ are not effected by a variation in $M_u$ of ±20%, but significant changes are seen when $M_u$ is varied more than ±10% in the errors of states $\bar{u}$, $\bar{w}$, $\bar{\theta}$, and $\bar{h}$.

$\underline{M_w}$. The dimensional variation of the M moment with speed w has a nominal value of -0.0111. The results of a numerical variation in a range of ±10% can be seen in Table 6. Since any alteration in the true value of $M_w$ has a strong effect on all the rms estimate errors, this derivative can be considered the most critical in the longitudinal motion estimation case.

$\underline{M_q}$. The dimensional variation of the pitching moment with pitch rate q has a nominal value of -0.924. The results of Table 7 on the rms estimate errors for a ±20% change in $M_q$ show that the sensitivity to variations in this parameter is minimal for all states.

$\underline{M_{\dot{w}}}$. The dimensional variation of the pitching moment with the rate of change of the downward speed w has a nominal value of -0.00051. Table 8 contains the rms errors data obtained by altering $M_{\dot{w}}$ ±20% from its nominal value. All the errors show a degree of sensitivity and the variation of errors is significant when $M_{\dot{w}}$ is changed by more than ±2%.

Since plots of the rms estimate errors versus changes in the particular dimensional derivatives for data identical to that of Tables 1-8 appears in [Ref. 4] in Figures 35-40, they will not be repeated in this work. A summary of the relative sensitivity of the rms estimate errors to changes in the individual dimensional derivatives follows in Table 9.

TABLE 5. RMS ESTIMATE ERRORS FOR LONGITUDINAL MOTION ESTIMATOR WITH VARIATION IN $M_u$ DERIVATIVE

| $M_u$ | $\bar{u}$ ft/s | $\bar{w}$ ft/s | $\bar{q}$ deg/s | $\bar{\theta}$ deg | $\bar{h}$ ft | $\bar{u}_g$ ft/s | $\bar{w}_g$ ft/s |
|---|---|---|---|---|---|---|---|
| -0.000943 | 2.234 | 5.061 | 0.416 | 0.305 | 6.230 | 4.775 | 5.689 |
| -0.000865 | 2.115 | 5.089 | 0.416 | 0.310 | 6.640 | 4.775 | 5.695 |
| -0.000825 | 2.104 | 5.094 | 0.416 | 0.314 | 7.531 | 4.776 | 5.698 |
| -0.000786* | 2.090 | 5.102 | 0.416 | 0.317 | 8.245 | 4.776 | 5.701 |
| -0.000747 | 1.993 | 5.105 | 0.416 | 0.318 | 8.595 | 4.777 | 5.703 |
| -0.000707 | 1.866 | 5.108 | 0.416 | 0.319 | 8.832 | 4.779 | 5.705 |
| -0.000629 | 1.566 | 5.111 | 0.416 | 0.322 | 9.178 | 4.783 | 5.708 |

TABLE 6. RMS ESTIMATE ERRORS FOR LONGITUDINAL MOTION ESTIMATOR WITH VARIATION IN $M_w$ DERIVATIVE

| $M_w$ | $\bar{u}$ ft/s | $\bar{w}$ ft/s | $\bar{q}$ deg/s | $\bar{\theta}$ deg | $\bar{h}$ ft | $\bar{u}_g$ ft/s | $\bar{w}_g$ ft/s |
|---|---|---|---|---|---|---|---|
| -0.01165 | 18.43 | 8.350 | 0.502 | 0.455 | 29.870 | 4.778 | 5.695 |
| -0.0113 | 5.163 | 5.142 | 0.433 | 0.373 | 17.790 | 4.778 | 5.694 |
| -0.0112 | 3.110 | 5.113 | 0.418 | 0.321 | 10.427 | 4.777 | 5.699 |
| -0.0111 * | 2.090 | 51.102 | 0.416 | 0.317 | 8.245 | 4.776 | 5.701 |
| -0.0109 | 5.206 | 5.018 | 0.419 | 0.325 | 16.650 | 4.776 | 5.700 |
| -0.01055 | 13.652 | 5.342 | 0.447 | 0.430 | 12.204 | 4.776 | 5.918 |
| -0.00999 | 19.13 | 18.95 | 0.475 | 0.704 | 68.98 | 4.756 | 6.102 |

TABLE 7. RMS ESTIMATE ERRORS FOR LONGITUDINAL MOTION ESTIMATOR WITH VARIATION IN $M_q$ DERIVATIVE

| $M_q$ | $\bar{u}$ ft/s | $\bar{w}$ ft/s | $\bar{q}$ deg/s | $\bar{\theta}$ deg | $\bar{h}$ ft | $\bar{u}_g$ ft/s | $\bar{w}_g$ ft/s |
|---|---|---|---|---|---|---|---|
| -1.109 | 2.091 | 5.102 | 0.416 | 0.316 | 8.230 | 4.776 | 5.701 |
| -1.016 | 2.091 | 5.102 | 0.416 | 0.316 | 8.234 | 4.776 | 5.701 |
| -0.970 | 2.091 | 5.102 | 0.416 | 0.317 | 8.236 | 4.776 | 5.701 |
| -0.924 * | 2.090 | 5.102 | 0.416 | 0.317 | 8.245 | 4.776 | 5.701 |
| -0.880 | 2.090 | 5.102 | 0.416 | 0.316 | 8.244 | 4.776 | 5.701 |
| -0.832 | 2.089 | 5.102 | 0.416 | 0.316 | 8.236 | 4.776 | 5.701 |
| -0.7392 | 2.089 | 5.102 | 0.416 | 0.316 | 8.232 | 4.776 | 5.701 |

TABLE 8. RMS ESTIMATE ERRORS FOR LONGITUDINAL MOTION ESTIMATOR WITH VARIATION IN $M_{\dot{w}}$ DERIVATIVE

| $M_{\dot{w}}$ | $\bar{u}$ ft/s | $\bar{w}$ ft/s | $\bar{q}$ deg/s | $\bar{\theta}$ deg | $\bar{h}$ ft | $\bar{u}_g$ ft/s | $\bar{w}_g$ ft/s |
|---|---|---|---|---|---|---|---|
| -0.00061 | 2.610 | 5.053 | 0.417 | 0.273 | 10.665 | 4.775 | 5.688 |
| -0.00056 | 2.476 | 5.089 | 0.417 | 0.295 | 6.301 | 4.776 | 5.703 |
| -0.00053 | 2.398 | 5.091 | 0.417 | 0.304 | 4.150 | 4.776 | 5.698 |
| -0.00051* | 2.090 | 5.102 | 0.416 | 0.317 | 8.245 | 4.776 | 5.701 |
| -0.00049 | 2.491 | 5.108 | 0.416 | 0.322 | 9.757 | 4.776 | 5.702 |
| -0.00046 | 2.976 | 5.118 | 0.415 | 0.332 | 12.067 | 4.776 | 5.703 |
| -0.00041 | 3.570 | 5.124 | 0.415 | 0.340 | 13.536 | 4.776 | 5.703 |

TABLE 9. RELATIVE SENSITIVITY OF THE RMS ESTIMATE ERRORS TO CHANGES IN DERIVATIVES

| DERIVATIVE | NS | RS | VS |
|---|---|---|---|
| $X_u$ | X | | |
| $X_w$ | X | | |
| $Z_u$ | | X | |
| $Z_w$ | | | X |
| $M_u$ | | X | |
| $M_w$ | | | X |
| $M_q$ | X | | |
| $M_{\dot{w}}$ | | X | |

NS = Not sensitive

RS = Relatively sensitive

VS = Very sensitive

### 3. Analysis of Longitudinal Motion Estimation After Augmentation

Including the distance measurement to the system required state augmentation of the system and measurement models as demonstrated by equations (30) and (31) that follow:

$$\begin{bmatrix} \dot{u} \\ \dot{w} \\ \dot{q} \\ \dot{\theta} \\ \dot{h} \\ \dot{u}_g \\ \dot{w}_g \\ \dot{d} \end{bmatrix} = \begin{bmatrix} -0.015 & 0.004 & 0 & -0.0322 & 0 & -0.015 & 0.004 & 0 \\ -0.074 & -0.806 & 0.824 & 0 & 0 & -0.074 & -0.806 & 0 \\ -0.749 & -10.7 & -1.344 & 0 & 0 & -0.749 & -10.7 & 0 \\ 0 & 0 & 1.0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -0.1 & 0 & 0.824 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -0.413 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -0.853 & 0 \\ 1.0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} u \\ w \\ q \\ \theta \\ h \\ u_g \\ w_g \\ d \end{bmatrix}$$

$$+ \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0.413 & 0 & 0 \\ 0 & 0.853 & 0 \\ 0 & 0 & 1.0 \end{bmatrix} \begin{bmatrix} \mu_u \\ \mu_w \\ \mu_d \end{bmatrix} \qquad (30)$$

and

$$\begin{bmatrix} z_q \\ z_h \\ z_d \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ w \\ q \\ \theta \\ h \\ u_g \\ w_g \\ d \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v_q \\ v_h \\ v_d \end{bmatrix} \qquad (31)$$

In these equations the state variables are u, velocity along the X axis, w, velocity along the Z axis, q, pitch rate, $\theta$, pitch angle, h, altitude and d, distance traveled along the X axis. The units are: u and w in 10 ft/s, q in 0.01 rad/s, $\theta$ in 0.01 rad, h in 100 ft, and d in 10 ft.

The OPTSYS program, when executed with the data from equations (30) and (31) above and the standard deviation values from Appendix B, yielded the following:

filter gain matrix K

| | | |
|---|---|---|
| 0.0592 | 0.0570 | 0.0014 |
| 0.2646 | -0.1611 | -0.0007 |
| 3.5168 | 0.0404 | 0.0002 |
| 0.0011 | -0.0810 | -0.0007 |
| 0.0135 | 0.1353 | 0.0001 |
| -0.0114 | 0.0356 | -0.0002 |
| -1.2876 | 0.1283 | 0.0005 |
| 0.0469 | 0.0561 | 1.0014 |

filter eigenvalues

-3.103 ± 4.1103

-1.000

-0.4146

-0.3152

-0.0485 ± 0.0548

-0.0551

rms estimate errors

| | | | | | |
|---|---|---|---|---|---|
| $\tilde{u}$ | = | 2.090 ft/s | $\tilde{\theta}$ | = | 0.316 deg |
| $\tilde{w}$ | = | 5.102 ft/s | $\tilde{h}$ | = | 8.225 ft |
| $\tilde{q}$ | = | 0.416 deg/s | $\tilde{u}_g$ | = | 4.776 ft/s |
| $\tilde{w}_g$ | = | 5.701 ft/s | $\tilde{d}$ | = | 38.730 ft |

The next step in the analysis was to perturb each directional derivative independently by specific amounts from its nominal value and to observe the effect on the rms estimate errors. This process was carried out for all eight directional derivatives and the response was the same for each case -- even a slight perturbation of -0.1% of any directional derivative from its nominal value caused the rms estimate errors to increase without bound. This behavior indicated that any incorrect implementation of dynamics in the new system formed by the augmentation of the distance measurement would cause instability and the Kalman Filter to diverge.

Several system parameters were individually modified and the analysis repeated in hopes of finding a stable system for which the

Kalman Filter converged. The coefficient for the distance term in the process noise distribution matrix was varied from 0.01-5.0, the power spectral density process noise entry for distance was changed in a range of 1.105-30.0, and the distance term for the power spectral density measurement noise adjusted over a range of 0.03-30.0. None of these trials led to a stable system.

A modal analysis was also performed using the open loop eigenvalues from the OPTSYS output listing. The system of equations (30) and (31) when transformed into modal coordinates give equations (32) and (33) below:

$$\begin{bmatrix} \dot{\xi}_E \\ \dot{\xi}_d \\ \dot{\xi}_{s1} \\ \dot{\xi}_{s2} \\ \dot{\xi}_{p1} \\ \dot{\xi}_{p2} \\ \dot{\xi}u_g \\ \dot{\xi}w_g \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1.076 & 2.957 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2.957 & -1.076 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -0.006 & 0.024 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.024 & -0.006 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -0.413 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -0.853 \end{bmatrix} \begin{bmatrix} \xi_E \\ \xi_d \\ \xi_{s1} \\ \xi_{s2} \\ \xi_{p1} \\ \xi_{p2} \\ \xi u_g \\ \xi w_g \end{bmatrix}$$

$$+ \begin{bmatrix} 0 & 0.1 & 0 \\ -1.0 & 0 & 1.0 \\ -0.028 & -0.088 & 0 \\ -0.110 & -3.153 & 0 \\ 1.027 & -0.048 & 0 \\ -0.210 & 1.467 & 0 \\ 0.420 & 0 & 0 \\ 0 & 1.836 & 0 \end{bmatrix} \begin{bmatrix} \mu_u \\ \mu_w \\ \mu_d \end{bmatrix} \qquad (32)$$

$$\begin{bmatrix} z_q \\ z_h \\ z_d \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1.0 & 0 & -0.0001 & 0.0005 & 0.0548 & 0.4802 \\ 1.0 & 0 & 0.0016 & 0.0016 & -0.0156 & -0.0612 & 0.0082 & -0.0025 \\ 0 & 1.0 & -0.0008 & -0.0004 & 1.0 & 0 & -0.0657 & -0.0252 \end{bmatrix} \begin{bmatrix} \xi_E \\ \xi_d \\ \xi_{s1} \\ \xi_{s2} \\ \xi_{p1} \\ \xi_{p2} \\ \xi u_g \\ \xi w_g \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v_g \\ v_h \\ v_d \end{bmatrix} \tag{33}$$

Consistent with the discussion in [Ref. 3], inspection of equation (32) revealed that the energy mode $\xi_E$ and the distance mode $\xi_d$, were neutrally stable (i.e., eigenvalues = 0). Inspection of equation (33) showed that $\xi_E$ was unobservable with $z_q$ and $z_d$ and that $\xi_d$ was unobservable with $z_q$ and $z_h$. Equation (32) also disclosed that $\xi_E$ was undisturbed by $u_g$ and d, and that $\xi_d$ was undisturbed by $w_g$. Therefore, destabilization was conducted in an attempt to prevent filter divergence. Both total and modal destabilization described earlier in this work and in [Ref. 3] were performed in amounts of 0.040 and 1.0 using the OPTSYS program. The filter gains computed for the destabilized system were then executed in the Sensitivity Covariance Program with each of the modified parameter combinations discussed earlier. Without exception, the rms estimation errors increased without bound when the least sensitive dimensional derivative $X_u$ was perturbed by as little as ±1%.

## V. CONCLUSIONS AND RECOMMENDATIONS

### A. CONCLUSIONS

The conclusions reached were based on the results obtained and will therefore be presented in three parts.

1. Motion Estimation Analysis for Exact Dynamics

The data in the results is in agreement with that of both [Ref. 3] and [Ref. 4]. It shows that both the Kalman Filters for initial longitudinal and lateral cases are stable when the true values for the system dynamics are implemented.

2. Longitudinal Motion Estimation Analysis

The results from Tables 1-8 and summarized in Table 9 are consistent with those of [Ref. 4]. The stability derivatives $Z_w$ and $M_w$ cause the strongest changes in the rms estimate errors when they are varied. Basically, $Z_w$ and $M_w$ must be quite accurately reproduced in the filter to prevent divergence. Changes in the stability derivatives $Z_u$, $M_u$, and $M_{\dot{w}}$ reflect intermediate variations in nearly all the rms estimate errors. For the model considered a tolerance of more than ±5% affects the accuracy in the radial position since large variations in $\bar{u}$ occur. A tollerance of perhaps ±20% can be accepted in the dimensional derivatives $X_u$, $X_w$, and $M_q$ for tnis model since no important effect is noted in the rms errors over that range.

3. Analysis of Longitudinal Motion Estimation After Augmentation

From the results presented earlier for the new system model fomed by the augmentation of a distance measurement and associated process and

measurement noise parameters, it is apparent that the corresponding Kalman Filter will diverge for even a slight variation in any of the dimensional derivatives from their nominal values. Even the Kalman Filter developed by system destabilization proved to be unstable with the parameters used.

B. RECOMMENDATIONS

Further analysis of the augmented system including the distance or position estimation is desirable. Perhaps a more in-depth study of the measurement parameter scaling would enable the development of a stable Kalman Filter for at least a destabilized system.

# VI. SUMMARY

The sensitivity analyses performed in this work have revealed the importance of accuracy in determining system dynamics utilized in formulating the model for the Kalman Filter. The relative sensitivity of the rms estimation errors to variance in each of the particular dimensional derivatives is shown in Table 9 for the Longitudinal Motion Estimator.

The longitudinal system augmented with the distance measurement developed appears to be extremely sensitive to variations in all the dimensional derivatives. Further analysis of the model developed is suggested.

# APPENDIX A

## LIST OF SYMBOLS

| Regular Symbols | Definition |
|---|---|
| A | Modal transformation of F matrix |
| B | Modal transformation of $\Gamma$ matrix |
| C | Modal transformation of H matrix |
| D | Dutch roll mode |
| d | Distance traveled along the X axis |
| E | Destabilization matrix |
| F | System dynamics matrix |
| f | Subscript for filter |
| F' | Destabilized matrix |
| g | Subscript for wind speed |
| H | Measurement matrix |
| H | Heading Mode |
| h | Altitude |
| INS | Inertial Navigation System |
| K | Kalman Filter gain matrix |
| L | Rolling moment (about X axis) |
| M | Pitching moment (about Y axis) |
| MDS | Modal destabilization |
| N | Yawing moment (about Z axis) |
| n | Non-white gaussian noise |
| P | Covariance propagation of the estimate error matrix |
| P | Perturbed roll rate |
| Q | Covariance matrix of w |
| q | Perturbed pitch rate |
| R | Covariance matrix of v |
| r | Perturbed yaw rate |
| S | Spiral mode |

| Regular Symbols | Definition |
|---|---|
| SKF | Steady-State Kalman Filter |
| T | Transformation matrix |
| UNS | Undisturbed neutrally stable |
| u | Perturbed forward speed (along X axis) |
| V | Forward velocity |
| v | Perturbed side velocity |
| w | Driving white gaussian noise |
| $w_g$ | Perturbed downward velocity |
| X | Reference axis |
| $\underline{x}$ | State vector of the system |
| $\hat{x}$ | State estimate vector |
| $\tilde{x}$ | Estimate error vector |
| Y | Reference axis |
| z | Measurement vector |
| Z | Reference Axis |

| Greek Symbols | Definition |
|---|---|
| $\psi$ | Heading angle |
| $\theta$ | Perturbed pitch attitude angle |
| $\phi$ | Perturbed bank (roll) angle |
| $\beta$ | Sideslip angle |
| $\Gamma$ | Driving noise matrix |
| $\sigma$ | Eigenvalue constrain |
| $\sigma$ | Standard deviation |
| $\xi$ | Transformed state vector |
| $\tau$ | Time |

APPENDIX B

## AERODYNAMIC DATA AND PROBABILISTIC INFORMATION

$v = 820$ ft/s

1. Longitudinal Model

a. Dimensional Derivatives

| | | |
|---|---|---|
| $X_u =$ | 0.015 | 1/s |
| $X_w =$ | 0.004 | 1/s |
| $Z_u =$ | -0.074 | 1/s |
| $Z_w =$ | -0.0806 | 1/s |
| $M_u =$ | -0.0786 | 1/s-ft |
| $M_w =$ | -0.0111 | 1/s-ft |
| $M_q =$ | -0.924 | 1/s-rad |
| $M_{\dot{w}} =$ | -0.00051 | 1/ft |

b. Distrubance Noise Standard Deviation

$\sigma_u = \sigma_w = 1.105$ 1/s (10 ft/s)²

$\sigma_x = 30.0$ 1/s (10 ft/s)²

(7 ft/s rms gust with a 930-ft correlation distance).

c. Observation Noise Standard Deviation

$\sigma_q = 0.15$ s (0.01 rad/s)²

$\sigma_h = 0.05$ s (100 ft)²

$\sigma_d = 30.0$ s (10 ft)²

## 2. Lateral Models

### a. Dimensional Derivatives

$Y_v = -0.0868$ 1/s

$N'_\beta = 2.14$ 1/s

$N'_r = -0.228$ 1/s

$N'_p = -0.0204$ 1/s

$L'_\beta = -4.41$ 1/s²

$L'_r = 0.334$ 1/s

$L'_p = -1.181$ 1/s

### b. Disturbance Noise Standard Deviation

$\sigma = 1.63 \times 10^{-4}$ 1/s

(7 ft/s rms gust with a 930-ft correlation distance)

### c. Observation Noise Standard Deviation

$\sigma_p = 1.5 \times 10^{-5}$ s

$\sigma_\psi = 1.5 \times 10^{-5}$ 1/s

## APPENDIX C

## AN AID TO USING OPTSYS AT NPS

### I. INTRODUCTION

One of the tasks involved in my thesis work at the Naval Postgraduate School (NPS) was to verify some of the data of reference [1] which investigated the sensitivity of the Steady-State Kalman Filters as lateral and longitudinal estimators in Strapdown Inertial Navigation Systems (INS). One of the recurring, essential calculations was for the steady-state gains of each system model considered. Fortunately, the OPTSYS computer program was available in Fortran at the computer center to help perform this enormous job. The use of the OPTSYS program was covered by reference [2], but not in adequate detail for easy application. After much trial-and-error, frustration, attempted decoding with the assistance of the computer center staff, and prayer, and at the expense of many man-hours of time, our Lord enabled me to properly fill out and order the data cards for a particular modeled system and obtain the expected results upon execution of the program. Since Professor Collins has several other students in need of a users working knowledge of the OPTSYS program and anyone using Kalman Filters can benefit as well, I am writing a more detailed description of how to correctly input data by discussing a specific example. The intent of this paper is to supplement the guidance of reference [2] and further facilitate research at NPS.

## II. MODEL AND ESTIMATION

Consider the linear time-invariant system given by

$$\dot{x} = Fx + \Gamma w$$

$$z = Hx + v$$

where x represents the states of the system; z is the measurement vector; F is the system matrix; $\Gamma$ is the driving noise coefficient matrix; H is the measurement scaling matrix; and w and v are independent, zero-mean, white gaussian noise processes with covariance matrices Q and R, respectively.

A continuous time Kalman Filter for this system is described by

$$\dot{\hat{x}} = F\hat{x} + K(z - H\hat{x})$$

where x is the state estimate and K is the matrix of the steady-state gains of the Kalman Filter. The implementation of the System Model and the Kalman Filter are shown below in Figure C-1 [Ref. 1].

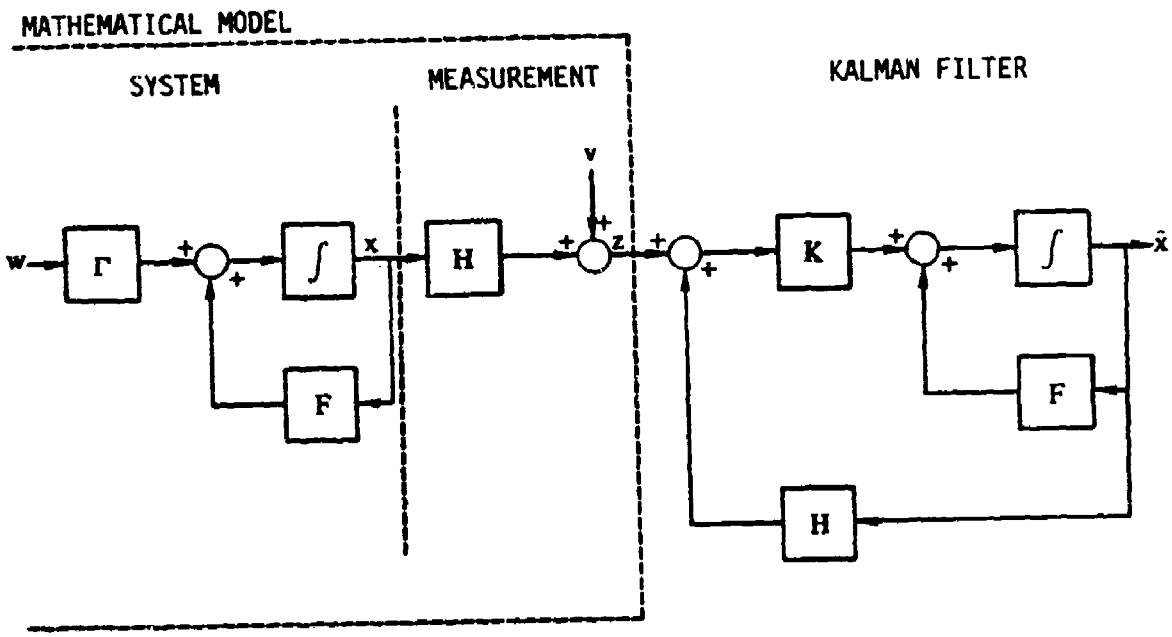


Figure C-1.  System Model and Kalman Filter

## III. AN EXAMPLE OF LONGITUDINAL MOTION ESTIMATION

After state vector augmentation, the resultant model of longitudinal motion of an aircraft of the form $\dot{x} = Fx + \Gamma w$ is

$$\begin{bmatrix} \dot{u} \\ \dot{w} \\ \dot{q} \\ \dot{\theta} \\ \dot{h} \\ \dot{u}_g \\ \dot{w}_g \end{bmatrix} = \begin{bmatrix} -0.015 & 0.004 & 0 & -0.0322 & 0 & -0.015 & 0.004 \\ -0.074 & -0.806 & 0.824 & 0 & 0 & -0.074 & -0.806 \\ -0.749 & -10.7 & -1.344 & 0 & 0 & -0.749 & -10.7 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & -0.1 & 0 & 0.0824 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -0.413 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -0.853 \end{bmatrix} \begin{bmatrix} u \\ w \\ q \\ \theta \\ h \\ u_g \\ w_g \end{bmatrix}$$

$$+ \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0.413 & 0 \\ 0 & 0.853 \end{bmatrix} \begin{bmatrix} \mu_u \\ \mu_w \end{bmatrix}$$

where the units are scaled such that $u$, $w$, $u_g$, and $w_g$ must be multiplied by 10 to give feet per second, $q$ by 0.01 to give radians per second, $\theta$ by 0.01 to give radians, and $h$ by 100 to give units of feet [Ref. 1].

The corresponding measurement model in the form $z = Hx + Iv$ is given by

$$\begin{bmatrix} z_q \\ z_h \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} u \\ w \\ q \\ \theta \\ h \\ u_g \\ w_g \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_q \\ v_h \end{bmatrix} \qquad (26)$$

For this model $Q_u = Q_w = 1.105\ (10\ \text{ft/s})^2/\text{s}$, $R_q = 0.15\ (0.01\ \text{rad/s})^2$ and $R_h = 0.05\ (100\ \text{ft})^2\ \text{s}$ [Ref. 3].

## IV. APPLYING OPTSYS TO THE EXAMPLE

The essential input data that will enable OPTSYS to calculate the steady-state gains of the Kalman Filter and many other parameters outlined in [Ref. 2] follows on page 55. The input data and control cards are described in the paragraphs below.

Card 1 - The 17 entries in every other column from column 2 through column 34 essentially tell OPTSYS what to compute. See [Ref. 2] for more details.

Card 2 - The 5 entries in every third column from 3 through 15 describe the system being modeled to OPTSYS. The first entry tells the number of states or order of the system-7 since there are seven rows in the F matrix. The second entry gives the number of controls-0 since u=0. The third entry tells that we have 2 measurements, while the fourth entry shows that two process noise sources exist. The fifth entry is always zero when filter synthesis is done. See [Ref. 2] if regulator synthesis only is desired.

Cards 3-16 - These cards contain the F matrix. The first six entries of each row go on one card with 12 columns for each entry-1-12, 13-24, ..., 60-72. The seventh entry for each row is placed in columns 1-12 of a continuation card that immediately follows the card with the first six entries of the row. Note that if our example system were 6x6, the F matrix would only take up cards 3-8.

The next three cards, 17-19 in our example, contain the H matrix. Note that this matrix is also entered on the cards by rows, but consecutively with an entry in every 12 columns with 6 entries per card as long as unused row elements remain! Thus the first entry of row 2 of the H matrix appears in columns 13-24 of card 18.

The next three cards, 20-22, hold the $\Gamma$ matrix. This matrix is also entered consecutively by rows with an entry in the first 14 groups of 12 columns on the cards!

The next to the last card gives the Q matrix. Note that this card has only the diagonal terms of the matrix in columns 1-12 and 13-24. See [Ref. 2] for matrices with non-diagonal terms.

The last card is for the R matrix and also has diagonal entries in columns 1-12 and 13-24. Again refer to [Ref. 2] if non-diagonal terms exist.

This supplement will be effective until the OPTSYS program is re-coded in WATFIV language. Its usage should greatly improve the efficienty and morale of those using the OPTSYS program on file at NPS Computer Center.

```
0 0 0 0 1 1 0 0 0 0 0 0 1 0 0 3 0
 7  0  2  2  0
      -0.015        0.004          0.0     -0.0322          0.0      -0.015
       0.004
      -0.074       -0.806        0.824         0.0           .0      -0.074
      -0.806
      -0.749  -1.07   E01       -1.344          .0           .0      -0.749
  -1.07   E01
          .0           .0          1.0          .0           .0          .0
          .0
          .0         -0.1           .0      0.0824           .0          .0
          .0
          .0           .0           .0          .0           .0      -0.413
          .0
          .0           .0           .0          .0           .0          .0
      -0.853
          .0           .0          1.0          .0           .0          .0
          .0           .0           .0          .0           .0         1.0
          .0           .0
          .0           .0           .0          .0           .0          .0
          .0           .0           .0          .0        0.413          .0
          .0        0.853
       1.105        1.105
        0.15         0.05
```

## REFERENCES


1. Matallana, J. A., "Sensitivity of the S.K.F. to Stability Derivatives Variations in an I.N.S.", Masters Thesis, Naval Postgraduate School, Monterey, California, 1980.

2. Walker R., "OPTSYS 4 at SCIP Computer Program", Stanford University, Aero/Astro Department, December 1979.

3. Bryson, A. E., Jr., "Kalman Filter Divergence and Aircraft Motion Estimators", Vol. 1, No. 1, AIAA Journal, January 1978.

# COMPUTER OUTPUTS

```
C/OPTSYS1 JOB (1480,0417),'POTTER G.G.'
C/ EXEC FORTCLG
C/SYSIN DD *
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION ACL(16,16),B(8,9),BA(16,16),CI(16),CR(16),CQ(16,16),
     *CWI(16),CWR(16),FBGC(8,16),FBGE(16,8),G(16,16),GM(16,16),
     *PRO(16,16),RC(8,8),SC(16,16),WR(32),WI(32),W11(16,16),
     *W21(16,16),X(32,32),GN(16,16),HO(8,16),D1(32),D2(32),RM(32,32),
     *Q(8,8),GAM(16,9),WNORM(16,16),WNORMI(16,16),DESTAB(16),
     *AA(16,16),BM(16,8),CM(8,16),D(8,8),DSTORE(16,16),
     *JCF(32),RES(32),AY(8),BB(32),CC(32),CP(16),GW(32,8),GV(32,8),
     *HY(8,32),HU(8,32)
      EQUIVALENCE (W11(1,1),GW(1,1)),(W11(1,1),GV(1,1)),
     1 (W21(1,1),HY(1,1)),(W21(1,1),HU(1,1))
      COMMON /PROG/ IOL,INC,IQ,IR,ISS,IM,ITF1,ITF2,ITF3,IFDFW,IE,
     *  IDSTAB,IDEBUG,ISET,IREG,IPSD,IYU,INORM
      DATA STAR/'*'/
    4 FORMAT(40I2)
  101 READ(5,4,END=100) IOL,IQ,INQ,IR,ISS,IM,ITF1,ITF2,ITF3,IFDFW,IE,
     *      IDSTAB,IDEBUG,ISET,IPSD,IYU,INORM
      READ(5,7432) NS,NC,NOB,NG,IREG
 7432 FORMAT(5I3)
      WRITE(6,4000) NS,NC,NOB,NG
 4000 FORMAT('1',2X,'ORDER OF SYSTEM =',I3,//,2X,'NUMBER OF CONTROLS ='
     1         ,I3,//,2X,'NUMBER OF OBSERVATIONS =',I3,//,2X,
     2          'NUMBER OF PROCESS NOISE SOURCES =',I3,//)
      N2=2*NS
      CALL INNER(NS,NC,NOB,NG,N2,ACL,B,BA,CI,CR,CQ,CWI,CWR,D,FBGC,FBGE,
     *G,GAM,GM,GN,HO,D1,D2,PRO,RM,RC,Q,SC,WR,WI,W11,W21,X,
     *WNORM,WNORMI,DESTAB,AA,BM,CM,JCF,RES,AY,BB,CC,CP,GW,GV,HY,HU,
     * DSTORE)
   99 READ(5,5,END=100) STA
    5 FORMAT(A2)
      IF(STAR.EQ.STA)  GOTO 101
      GOTO 99
  100 STOP
      END
      SUBROUTINE SETUP(BA,G,GAM,NS,NC,NG)
      RETURN
      END
      SUBROUTINE
     *      INNER(NS,NC,NO,NG,N2,ACL,B,BA,CI,CR,CQ,CWI,CWR,D,FBGC,FBGE,
     *G,GAM,GM,GN,HO,D1,D2,PRO,RM,RC,Q,SC,WR,WI,W11,W21,X,
     *WNORM,WNORMI,DESTAB,AA,BM,CM,JCF,RES,AY,BB,CC,CP,GW,GV,HY,HU,
     * DSTORE)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION ACL(NS,NS),B(NC,NC),BA(NS,NS),CI(NS),CR(NS),CQ(NS,NS),
     *CWI(NS),CWR(NS),FBGC(NC,NS),FBGE(NS,NO),G(NS,NS),GM(NS,NS),
     *      PRO(NS,NS),RC(NO,NO),SC(NS,NS),WR(N2),WI(N2),W11
     *(NS,NS),W21(NS,NS),X(N2,N2),GN(NS,NS),HO(NO,NS),D1(N2),D2(N2),RM(
     *N2,N2),Q(NG,NG),D(NO,NC),GAM(NS,NG),WNORM(NS,NS),WNORMI(NS,NS)
     *  ,DESTAB(NS),AA(NS,NS),BM(NS,NC),CM(NO,NS),JCF(N2),RES(N2),AY(NO
     *     ),BB(N2),CC(N2),CP(NS),GW(N2,NG),GV(N2,NO),HY(NO,N2),HU(NC,N2)
     *,DSTORE(NS,NS)
      COMMON /PROG/ IOL,INQ,IQ,IR,ISS,IM,ITF1,ITF2,ITF3,IFDFW,IE,
     *      IDSTAB,IDEBUG,ISET,IREG,IPSD,IYU,INORM
      REAL*8 FMT(20)
      NSQ=NS*NS
C
C***OUTPUT OPTIONS
C---IOL=1 IF THE OPEN LOOP EIGENSYSTEM IS DESIRED--OTHERWISE IOL=0
C---IQ=1 IF THE RMS VALUES OF THE CONTROL AND STATE ARE TO BE FOUND
C---INQ=1 IF ONLY B AND R ARE DIAGONAL
C   INQ=0 IF A, B, Q, AND R ARE DIAGONAL
C---IR=0 IF OPTIMAL FILTER AND REGULATOR EIGENSYSTEMS ARE TO BE FOUND
C   IR=1 IF EXTERNAL C MATRIX IS SUPPLIED
C   IR=2 IF EXTERNAL K IS SUPPLIED
C   IR=3 IF EXTERNAL C AND K ARE SUPPLIED
C---ISS=1 IF STEADY STATE VALUES ARE TO BE DETERMINED
C   IM=1 IF MODAL STATES DESIRED
C****
C
```

```
    3 FORMAT(' ','OPEN LOOP DYNAMICS MATRIX....',//)
 7444 FORMAT(6E12.5)
 6000 FORMAT(10(2X,1PD10.3),/,2X,10(2X,1PD10.3))
 6001 FORMAT('0',//,2X,'THE CONTROL DISTRIBUTION MATRIX....',//)
 6003 FORMAT('0',//,2X,'THE CONTROL WEIGHTING MATRIX.......',//)
 6010 FORMAT('0',//,2X,'PROCESS NOISE DISTRIBUTION MATRIX....',//)
 6011 FORMAT('0',//,2X,'POWER SPECTRAL DENSITY - PROCESS NOISE....',//)
 6051 FORMAT('0',//,2X,'MEASUREMENT SCALING MATRIX.....',//)
 6052 FORMAT('0',//,2X,'POWER SPECTRAL DENSITY - MEASUREMENT NOISE..',//
 6012 FORMAT('0',//,2X,'DIAGONAL OUTPUT COST MATRIX...',//)
 6013 FORMAT('0',//,2X,'OUTPUT COST MATRIX...',//)
 8515 FORMAT('0',//,2X,'MEASUREMENT FEEDTHROUGH MATRIX...',//)
      MH = NS
      M = N2
C
C  SUBROUTINE CHECK CHECKS THE CONSISTENCY OF REQUESTED OPTIONS
C
      CALL CHECK(EPS,NC,NG,NO)
      IF(ISET.EQ. 1)GO TO 9015
      DO 9010 I=1,NS
 9010 READ(5,7444)  (BA(I,J),J=1,NS)
      IF(IDSTAB .EQ. 0) GO TO 9014
      READ(5,7444)  (DESTAB(I),I=1,NS)
 9014 CONTINUE
      GO TO 9016
 9015 CALL SETUP(BA,G,GAM,NS,NG,NC)
 9016 CONTINUE
      WRITE(6,3)
      DO 5001 I= 1,NS
 5001 WRITE(6,6000)  (BA(I,J),J=1,NS)
      IF(IDSTAB .EQ. 0) GO TO 3999
      WRITE(6,480)
  480 FORMAT(/,' DESTABILIZATION CASE....',/,
     *  '    THE FOLLOWING VALUES WILL BE ADDED DOWN THE DIAGONAL TO ',
     *  'DESTABILIZE THE',/,'  ABOVE MATRIX. OPTIMAL GAINS FOR THE ',
     *  'DESTABILIZED SYSTEM ARE THEN USED',/,
     *  '    AS FIXED SUBOPTIMAL GAINS WITH THE ABOVE SYSTEM',/)
      WRITE(6,6000)  (DESTAB(I),I=1,NS)
 3999 CONTINUE
C***EIGENSYSTEM OF THE OPEN LOOP DYNAMICS
      IF (IOL.EQ.0.AND.IQ.EQ.0) GO TO 500
      IF (IOL.EQ.0.AND.NC.NE.0) GO TO 500
      DO 51 I = 1,NS
      DO 51 J = 1,NS
   51 GN(I,J) = BA(I,J)
C*********************************************
      CALL BALANC(NS,NS,GN,LOW,IHIGH,D1)
      CALL ORTHES(NS,NS,LOW,IHIGH,GN,D2)
      CALL ORTRAN(NS,NS,LOW,IHIGH,GN,D2,SC)
      CALL HQR2(NS,NS,LOW,IHIGH,GN,CWR,CWI,SC,IERR)
      IF(IERR .NE. 0) CALL EREXIT(NS,GN,IERR)
      CALL BALBAK(NS,NS,LOW,IHIGH,D1,NS,SC)
C*********************************************
C
C NORMALIZE AND PRINT OPEN LOOP EIGENSYSTEM
C
      IWRITE = 1
      CALL CNORM(CWR,CWI,SC,NS,IWRITE,NSQ,DDD,D1,D2,WNORM,WNOPMI,HO,CM,
     *      NO,NS)
      IF(IOL.EQ.2) RETURN
      IF (IQ.EQ.0.OR.(NC.NE.0.OR.IDSTAB.GT.0)) GO TO 500
      DO 496 I=1,NS
      IF (CWR(I).LT.0.) GO TO 496
      WRITE (6,495)
  495 FORMAT (///' PROGRAM TERMINATING DUE TO UNSTABLE SYSTEM')
      RETURN
  496 CONTINUE
      IF(IOL .EQ. 3) GO TO 510
      DO 497 I=1,NS
      DO 497 J=1,NS
  497 W11(I,J)=SC(I,J)
      CALL MINV (NSQ,W11,NS,DDD,D1,D2)
```

```
  500 CONTINUE
      IF(IDSTAB .EQ. 0) GO TO 510
C     ------------ FORM W * DIAG(DESTAB) * W-INV
      DO 505 J=1,NS
      DO 505 I=1,NS
  505 AA(I,J) = WNORM(I,J)*DESTAB(J)
      DO 507 I=1,NS
      DO 507 J=1,NS
        DDD = 0.D0
        DO 506 K=1,NS
  506   DDD = DDD + AA(I,K)*WNORMI(K,J)
      DSTORE(I,J) =DDD
  507 BA(I,J) = BA(I,J) + DDD
  510 CONTINUE
      READ(5,7444) ((HO(I,J),J=1,NS),I=1,NO)
      WRITE(6,6051)
      DO 1806 I = 1,NO
 1806 WRITE(6,6000) (HO(I,J),J=1,NS)
      IF(IM.NE.1) GO TO 8504
 8520 CALL MODE(WNORM,HO,CM,NS,NO,NS,2)
 8504 CONTINUE
      IF(IFDFW .EQ. 0) GO TO 8519
      READ(5,7444) ((D(I,J),J=1,NC),I=1,NO)
      WRITE(6,8515)
      DO 8517 I=1,NO
 8517 WRITE(6,6000) (D(I,J),J=1,NC)
 8519 CONTINUE
      NOB = 0
      IF (NC .EQ. 0) GO TO 1801
      IF(IOL .EQ. 3) GO TO 9035
      IF(IR.NE.1.AND.IR.NE.3) GOTO 9120
      IF(ISET.EQ.1) GO TO 9510
      READ(5,7444) ((G(I,J),J = 1,NC),I=1,NS)
 9510 CONTINUE
      READ(5,7444) ((FBGC(I,J),J=1,NS),I=1,NC)
      WRITE(6,6001)
      DO 9210 I=1,NS
 9210 WRITE(6,6000) (G(I,J),J=1,NC)
      IF(IM.NE.1) GO TO 8500
      CALL MODE(WNORMI,G,BM,NS,NS,NC,0)
 8500 CONTINUE
      GOTO 8399
 9120 DO 9020 I=1,NS
      DO 9020 J=1,NS
 9020 RM(I+MH,J)=0.0
      IF(INO .EQ. 1) GO TO 9215
      READ(5,7444) (AY(I),I=1,NO)
      DO 9505 I=1,NO
      DO 9505 J=1,NS
 9505 AA(I,J)= HO(I,J)*AY(I)
      GO TO 9219
 9215 READ(5,7444) ((Q(I,J),J=1,NO),I=1,NO)
      DO 9217 I=1,NO
      DO 9217 J=1,NS
      DDD = 0.D0
        DO 9216 K=1,NO
 9216   DDD = DDD + Q(I,K)*HO(K,J)
 9217 AA(I,J) = DDD
      WRITE(6,6013)
      DO 9218 I=1,NO
 9218 WRITE(6,6000) (Q(I,J),J=1,NO)
 9219 DO 9508 I=1,NS
      DO 9508 J=1,NS
      DO 9508 K=1,NO
 9508 RM(I+MH,J)=RM(I+MH,J) + AA(K,I)*HO(K,J)
 9030 IF(ISET.EQ.1) GO TO 9520
 9035 READ(5,7444) ((G(I,J),J = 1,NC),I=1,NS)
      IF(IOL .EQ. 3) GO TO 9045
 9520 CONTINUE
      DO 9040 I=1,NC
      DO 9040 J=1,NC
 9040 B(I,J)=0.0
```

```
      READ(5,7444) (B(I,I),I=1,NC)
      IF(INQ .EQ. 1) GO TO 9045
      WRITE (6,6012)
      WRITE (6,6000) (AY(I),I=1,ND)
 9045 WRITE (6,6001)
      DO 606 I = 1,NS
  606 WRITE (6,6000) (G(I,J),J = 1,NC)
      IF(IM.NE.1) GO TO 8501
      CALL MODE(WNORMI,G,BM,NS,NS,NC,0)
 8501 CONTINUE
      IF(IOL .EQ. 3) GO TO 8505
      WRITE (6,6003)
      DO 607 I=1,NC
  607 WRITE (6,6000) (B(I,J),J=1,NC)
 8399 IF(ITF1 .EQ. 0) GO TO 8400
C
C  OPEN LOOP TRANSFER FUNCTIONS
C
 8505 WRITE (6,9220)
 9220 FORMAT('0',//,2X,'OPEN LOOP TRANSFER FUNCTIONS...')
      ITFX=1
      CALL TF(NS,NS,NSQ,BA,AA,NC,G,BM,NO,HD,CM,IFDFW,D,BB,CC,CP,
     *     WR,WI,CWR,CWI,SC,JCF,RES,D1,D2,DOD,EPS,ITF1,ITFX)
 8400 IF(IOL .NE. 3) GO TO 8502
      IF(NG .EQ. 0) RETURN
      GO TO 625
 8502 CONTINUE
      IF(IR .EQ. 1 .OR. IR .EQ. 3) GO TO 9130
C*****************************************
C***CALCULATION OF CONTROL GAINS: FORMATION OF CONTROL HAMILTONIAN
C
C   **                         **          ***F AND FT ARE THE OPEN LOOP
C   *                           *             DYNAMICS MATRIX AND TRANSPOSE
C   *      F          -GM*BI*GMT*          ***BI IS NCXNC CONTROL WEIGHTING
C   *                           *             MATRIX
C   *                           *          ***A IS THE NSXNS STATE WEIGHTIN
C   *                           *             MATRIX
C   *     -A               -FT  *
C   **                         **          ***GM IS THE NSXNC CONTROL
C                                             DISTRIBUTION MATRIX
C********************************************************************
      DO 24 I = 1,NC
      DO 24 J = 1,MH
   24 PRO(I,J) =  G(J,I)/B(I,I)
      DO 25 I = 1,MH
      DO 25 J = 1,MH
      RM(I,J+MH)=0.D0
      DO 25 K = 1,NC
   25 RM(I,J+MH) = RM(I,J+MH) - G(I,K)*PRO(K,J)
C***2NX2N HAMILTONIAN MATRIX**************
C---DIAGONAL BLOCKS---M11 AND M22
      DO 26 I = 1,MH
      DO 26 J = 1,MH
      RM(I,J) =  BA(I,J)
      RM(I+MH,J+MH) = -BA(J,I)
C---M21 BLOCK
   26 RM(I+MH,J) =-RM(I+MH,J)
C---M12 BLOCK IS DEFINED IN LINE 25 ABOVE
C*********************************
   50 CONTINUE
      IF(IDEBUG .EQ. 0) GO TO 1050
      WRITE(6,6014)
 6014 FORMAT(//,' EULER-LAGRANGE SYSTEM MATRIX...',//)
      CALL RAPRNT(M,M,M,9,RM,4,'(9(1X,1PD13.6))')
 1050 CALL BALANC(M,M,RM,LOW,IHIGH,D1)
      CALL ORTHES(M,M,LOW,IHIGH,RM,D2)
      CALL ORTRAN(M,M,LOW,IHIGH,RM,D2,X)
      CALL HQR2(M,M,LOW,IHIGH,RM,WR,WI,X,IERR)
      IF(IERR .NE. 0) CALL EREXIT(M,RM,IERR)
      CALL BALBAK(M,M,LOW,IHIGH,D1,M,X)
C*********************************
C DEBUG DIAGNOSTICS ON E-L EQ
```

```
      IF(IDEBUG .EQ. 0) GO TO 53
      WRITE(6,9115)
 9115 FORMAT(//' EIGVAL AND EIGNVEC OF 2*N E-L EQ. AFTER HQR2'//)
      DO 52 I=1,N
   52 WRITE(6,9116) WR(I),WI(I)
 9116 FORMAT(1X,1P2D13.6)
      WRITE(6,9117)
 9117 FORMAT('0')
      CALL MAPRNT(N,N,N,9,X,4,'(9(1X,1PD13.6))')
   53 CONTINUE
      IF(IDSTAB .EQ. 1) GO TO 54
      IF (NOB.EQ.0) WRITE(6,9119)
      IF (NOB.NE.0) WRITE(6,9121)
   54 IF (NOB.NE. 0) GO TO 60
 9119 FORMAT('0',//,2X,'EIGENSYSTEM OF OPTIMAL CLOSED LOOP SYSTEM..',//
 9121 FORMAT('0',//,2X,'EIGENSYSTEM OF ESTIMATE ERROR EQUATION.....',//
            CALL RGAIN(N,NS,NC,NOB,WR,WI,X,GN,W11,RM,
     1W21,D1,CWR,CWI,SC,NNS,D2)
C CHECK EIGVEC
      IF(IDEBUG .EQ. 0) GO TO 750
      WRITE(6,9125)
 9125 FORMAT(' EIGENVECTORS FROM RGAIN PRIOR TO CNORM')
      CALL MAPRNT(NS,NS,NS,9,SC,4,'(9(1X,1PD13.6))')
  750 CONTINUE
C
C RESET FLAG AND F MATRIX FOR ITERATIVE DESTABILIZATION CASE
C
      IF(IDSTAB .EQ. 0) GO TO 9136
      DO 9135 I=1,NS
 9135 BA(I,I) = BA(I,I) - DESTAB(I)
      IR=1
 9136 CONTINUE
C  CALCULATION OF FEEDBACK GAIN
C
C**FEEDBACK GAINS---U = -(BINVERSE)*GT*GN
C---CALCULATE GT
      DO 801 I = 1,NC
      DO 801 J = 1,NS
      PRO (I,J) = 0.D0
      DO 800 K = 1,MM
  800 PRO (I,J) = PRO(I,J) +G(K,I)*GN(K,J)
  801 FBGC(I,J) = -PRO(I,J)            /B(I,I)
      IF(IDSTAB .EQ. 1) GO TO 9130
C
C NORMALIZE AND PRINT OPT. REG. CLOSED LOOP EIGENSYSTEM
C
      IWRITE =2
      CALL CNORM(CWR,CWI,SC,NS,IWRITE,NSQ,DDD,D1,D2,WNORM,WNORMI,FBGC,
     *     AA,NC,NS)
C**THE OPTIMUM FEEDBACK CONTROL GAINS
 9130 WRITE(6,977)
  977 FORMAT(///,' ','THE CONTROL GAINS ARE:',//)
      DO 968 I = 1,NC
  968 WRITE(6,978) (FBGC(I,J),J = 1,NS)
  978 FORMAT(' ',2X,1P6D14.6,/,2X,6D14.6)
C COMPUTE MODAL C MATRIX
C OPEN LOOP U-INVE SAVED IN WNORMI
      IF(IR .NE. 1) GO TO 985
C IN COMPUTING MODAL C RECOMPUTE U OPEN LOOP
C  SINCE WNORM USED TO STORE U AND U-INV FOR CLOSED LOOP SYSTEMS, AND
C  WNORMI USED TO SAVE U-INV OPEN LOOP
C
      DO 8510 I=1,NS
      DO 8510 J=1,NS
 8510 WNORM(I,J) = WNORMI(I,J)
      CALL MTINV(NSQ,WNORM,NS,DDD,D1,D2)
      CALL MODC(WNORM,FBGC,AA,NS,NC,NS,3)
  985 CONTINUE
C**THE CLOSED LOOP DYNAMICS MATRIX
      DO 160 I = 1,NS
      DO 160 J = 1,NS
      SUM      = 0.D0
```

```
      DO 150 K = 1,NC
  150 SUM=SUM               +G(I,K)*FBGC(K,J)
  160 ACL(I,J)=BA(I,J)+SUM
      WRITE(6,170)
  170 FORMAT('0','THE CLOSED LOOP DYNAMICS MATRIX IS..',//)
      CALL RAPRNT(MM,NE,MM,5,ACL,4,'(5(1X,1PD13.6))')
      IF(IR.NE.1.AND.IR.NE.3) GOTO 1801
      DO 9140 I=1,NS
      DO 9140 J=1,NS
 9140 GN(I,J)=ACL(I,J)
C*********************************************
      CALL BALANC(NS,NS,GN,LOW,IHIGH,D1)
      CALL ORTHES(NS,NS,LOW,IHIGH,GN,D2)
      CALL ORTRAN(NS,NS,LOW,IHIGH,GN,D2,SC)
      CALL HQR2(NS,NS,LOW,IHIGH,GN,CWR,CWI,SC,IERR)
      IF(IERR .NE. 0) CALL EREXIT(NS,GN,IERR)
      CALL BALBAK(NS,NS,LOW,IHIGH,D1,NS,SC)
C*********************************************
C
C NORMALIZE AND PRINT CLOSED LOOP SUBOPT. REG. EIGENSYSTEM
C
      IWRITE =3
      CALL CNORM(CWR,CWI,SC,NS,IWRITE,NSQ,DDD,D1,D2,WNORM,WNORMI,FBGC,
     *     AA,NC,NS)
      DO 9300 I=1,NS
      IF(CWR(I).LT.0.0) GOTO 9300
      WRITE(6,9310)
 9310 FORMAT(///,'     PROGRAM TERMINATING DUE TO UNSTABLE CLOSED LOOP
     *SYSTEM')
      RETURN
 9300 CONTINUE
      IF(IQ.NE.1) GOTO 1801
      DO 9400 I=1,NS
      DO 9400 J=1,NS
 9400 W11(I,J)=SC(I,J)
      CALL MINV(NSQ,W11,NS,DDD,D1,D2)
 1801 NCB=NO
      IF( NG .EQ. 0) RETURN
  625 IF(ISET.EQ. 1) GO TO 630
      READ(5,7444)   ((GAM(I,J),J=1,NG),I=1,NS)
  630 CONTINUE
      IF(IQL .EQ. 3) GO TO 9060
      IF(INQ.NE.0) GOTO 9070
      DO 9050 I=1,NG
      DO 9050 J=1,NG
 9050 Q(I,J)=0.0
      READ(5,7444)  (Q(I,I),I=1,NG)
      GOTO 9060
 9070 READ(5,7444)   ((Q(I,J),J=1,NG),I=1,NG)
 9060 WRITE(6,6010)
      DO 626 I = 1,NS
  626 WRITE(6,6000)  (GAM(I,J),J=1,NG)
      IF(IN.NE.1) GO TO 8503
      CALL MODE(WNORMI,GAM,AA,NS,NS,NG,1)
 8503 CONTINUE
      IF(IQL .EQ. 3) RETURN
      WRITE(6,6011)
      DO 627 I = 1,NG
  627 WRITE(6,6000)  (Q(I,J),J=1,NG)
  628 IF((IQ.EQ.0).AND.(NG.EQ.0)) GOTO 389
      DO 378 I = 1,NG
      DO 378 J = 1,NS
      PRO(I,J) = 0.D0
      DO 378 K = 1,NG
  378 PRO(I,J) = PRO(I,J)+Q(I,K)*GAM(J,K)
      DO 379 I = 1,NS
      DO 379 J = 1,NS
      CQ(I,J) = 0.D0
      DO 379 K = 1,NG
  379 CQ(I,J) = CQ(I,J)-GAM(I,K)*PRO(K,J)
 9100 IF(IREG .EQ. 1) GO TO 8505
C**CALCULATION OF FILTER GAINS:FORMATION OF ESTIMATION HAMILTONIAN
```

```
C
C    **                                  **          ***F AND FT ARE SAME AS FOR
C    *                                    *             CONTROL HAMILTONIAN
C    *      F               -GM*Q*GMT*    *        ***Q IS NGXNG STATE DISTURBANCE
C    *                                    *               RCOVARIANCE
C    *                                    *          ***R IS NOXNO MEASUREMENT NOISE
C    *                                    *               RCOVARIANCE
C    *  -HOT*RIN*HO         -FT           *          ***HO IS NOXNS MEASUREMENT MATRI
C    **                                  **          ***GM IS NSXNG STATE DISTURBANCE
C                                                          DISTRIBUTION MATRIX
C**********************************************************************************
 1805 DO 9110 I=1,NO
      DO 9110 J=1,NO
 9110 RC(I,J)=0.0
      READ(5,7444) (RC(I,I),I=1,NO)
      WRITE(6,6052)
      DO 1807 I = 1,NO
 1807 WRITE(6,6000) (RC(I,J),J=1,NO)
 8506 IF(ITF2 .EQ. 0) GO TO 28
C
C  NOISE TRANSFER FUNCTIONS
C
      WRITE(6,9230)
 9230 FORMAT('0',//,2X,'NOISE TRANSFER FUNCTIONS '
     *      ,'THROUGH THE CLOSED LOOP SYSTEM..')
      ITFI=2
      IZERO=0
      CALL TF(NS,NS,NSQ,ACL,AA,NG,GAM,BM,NO,HO,CM,IZERO,D,BB,CC,CP,
     *     WR,WI,CWR,CWI,SC,JCP,RES,D1,D2,DDD,EPS,ITF2,ITFX)
      IF(IREG .EQ. 1) RETURN
   28 CONTINUE
      IF(IREG .EQ. 1) GO TO 388
      IF(IR.LT.2) GOTO 1811
      READ(5,7444) ((FBGE(I,J),J=1,NO),I=1,NS)
      GOTO 9320
 1811 CONTINUE
C*** THE MEASUREMENT MATRIX
C** (HOT*RIN*HO===>SC
      DO 30 I = 1,NO
      DO 30 J = 1,MH
   30 PRO(I,J) = HO(I,J)/RC(I,I)
      DO 31 I = 1,MH
      DO 31 J = 1,MH
      RM(I+MH,J) = 0.D0
      DO 31 K = 1,NO
   31 RM(I+MH,J) = RM(I+MH,J) - HO(K,I)*PRO(K,J)
C***GM*Q*GMT===>CC
      DO 35 I = 1,NS
      DO 35 J = 1,NS
      RM(I,J)=BA(I,J)
      RM(I+MH,J+MH)=-BA(J,I)
   35 RM(I,J+NS) =CQ(I,J)
      GO TO 50
C***GO BACK TO 50 TO SET UP THE FILTER HAMILTONIAN
C    AND CALCULATE THE FILTER GAINS
 60             CALL RGAIN(M,NS,NC,NOB,WR,WI,X,GN,GM ,RM,
     1W21,D1,CR,CI,PRO ,MHS,D2)
C CHECK EIGVEC
      IF(IDEBUG .EQ. 0) GO TO 58
      WRITE(6,9125)
      CALL AAPPNT(NS,NS,NS,9,PRO,4,'(9(1X,1PD13.6))')
   58 CONTINUE
      IF(IDSTAB .EQ. 1) GO TO 9311
C
C NORMALIZE AND PRINT OPT. ESTIMATOR EIGENSYSTEM
C
      IWRITE=4
      CALL CNOPM(CR,CI,PRO,NS,IWRITE,NSQ,DDD,D1,D2,WNOPM,WNOPMI,HO,AA,
     *     NC,NS)
 9311 DO 61 I= 1,MH
      DO 61 J= 1,NO
   61 PRO(I,J) = -HO(J,I)/RC(J,J)
```

```
      DO 62 I = 1,NH
      DO 62 J = 1,NO
      FBGE(I,J) =0.D0
      DO 62 K = 1,NH
   62 FBGE(I,J) = FBGE(I,J)+GN(I,K)*PRO(K,J)
      IF(IDSTAB .EQ. 1) GO TO 9320
      WRITE(6,1501)
      CALL RAPRNT(NH,NH,NH,5,GN,4,'(5(1X,1PD13.6))')
      WRITE(6,1510)
      DO 9312 I=1,NH
 9312 X(I,I) = DSQRT(GN(I,I))
      WRITE(6,1520) (X(I,I),I=1,NH)
 9320 WRITE(6,1018)
 1018 FORMAT('0','FILTER STEADY STATE GAINS......',//)
      DO 63 I = 1,NH
   63 WRITE(6,1019) (FBGE(I,J),J =1,NO)
 1019 FORMAT(' ',2X,1P6D14.6)
C COMPUTE MODAL K MATRIX
C OPEN LOOP U-INV SAVED IN WNORMI
      IF(IM .NE. 1) GO TO 9330
      CALL MODE(WNORMI,FBGE,AA,NH,NH,NO,4)
 9330 CONTINUE
C
C RESET FLAG AND F MATRIX FOR ITERATIVE DESTABILIZATION CASE
C
      IF(IDSTAB .EQ. 0) GO TO 9338
      DO 9335 I=1,NS
      DO 9335 J=1,NS
 9335 BA(I,J) = BA(I,J)-DSTORE(I,J)
      IR=2
 9338 CONTINUE
      DO 9340 I=1,NS
      DO 9340 J=1,NS
      SUM=0.0
      DO 9350 K=1,NO
 9350 SUM=SUM+FBGE(I,K)*HO(K,J)
 9340 PRO(I,J)=BA(I,J)-SUM
      WRITE(6,9361)
 9361 FORMAT('0','THE CLOSED LOOP FILTER DYNAMICS MATRIX IS..',//)
      CALL RAPRNT(NS,NS,NS,5,PRO,4,'(5(1X,1PD13.6))')
      IF(IR .LT. 2) GO TO 9500
C******************************************
      CALL BALANC(NS,NS,PRO,LOW,IHIGH,D1)
      CALL ORTHES(NS,NS,LOW,IHIGH,PRO,D2)
      CALL ORTRAN(NS,NS,LOW,IHIGH,PRO,D2,GM)
      CALL HQR2(NS,NS,LOW,IHIGH,PRO,CR,CI,GM,IERR)
      IF(IERR .NE. 0) CALL EREXIT(NS,PRO,IERR)
      CALL BALBAK(NS,NS,LOW,IHIGH,D1,NS,GM)
C******************************************
      WRITE(6,9121)
C
C NORMALIZE AND PRINT SUBOPT. ESTIMATOR EIGENSYSTEM
C
      IWRITE=5
      CALL CNORM(CR,CI,GM,NS,IWRITE,NSQ,DDD,D1,D2,WNORM,WNORMI,HO,AA,
     *     NO,NS)
      DO 9410 I=1,NS
      IF(CR(I).LT.0.0) GOTO 9410
      WRITE(6,9420)
 9420 FORMAT(////,' PROGRAM TERMINATING DUE TO UNSTABLE FILTER')
      RETURN
 9410 CONTINUE
      GO TO 9501
 9500 IF(IQ.EQ.0) GO TO 389
 9501 DO 65 I = 1,NO
      DO 65 J = 1,NH
      PRO(I,J) = 0.D0
      DO 65 K = 1,NO
   65 PRO(I,J) = PRO(I,J)+RC(I,K)*FBGE(J,K)
      DO 66 I = 1,NH
      DO 66 J = 1,NH
      CQ(I,J) = 0.D0
```

```
      DO 66 K = 1,NO
   66 CQ(I,J) = CQ(I,J)-FBGE(I,K)*PRO(K,J)
  398 CONTINUE
C***THE RMS STATE AND CONTROL RESPONSES
      IR=IR+1
      GOTO (9360,9360,9380,9380), IR
 9380 DO 9705 I=1,NS
      DO 9705 J=1,NG
      X(I,J)=0.0
      DO 9705 K=1,NG
 9705 X(I,J)=X(I,J)+GAM(I,K)*Q(K,J)
      DO 9710 I=1,NS
      DO 9710 J=I,NS
      SUM=0.0
      DO 9711 K=1,NG
 9711 SUM=SUM-X(I,K)*GAM(J,K)
      PRO(I,J)=SUM+CQ(I,J)
      PRO(J,I)=PRO(I,J)
      CQ(I,J)=SUM
      CQ(J,I)=SUM
      W21(I,J)=GM(I,J)
 9710 W21(J,I)=GM(J,I)
      CALL MINV(NSQ,W21,NS,DDD,D1,D2)
      CALL SCOV (NS,GM,W21,CR,CI,NS,GM,W21,CR,CI,PPO,GN)
      WRITE(6,1501)
 1501 FORMAT('0',//,2X,'THE COVARIANCE OF THE ESTIMATION ERROR',//)
      CALL RAFBNT(MM,MM,MM,5,GN,4,'(5(1X,1PD13.6))')
      WRITE(6,1510)
 1510 FORMAT('0',/,2X,'RMS VALUES OF THE ESTIMATION ERROR',/)
      DO 9715 I=1,MM
 9715 X(I,I) = DSQRT(GN(I,I))
      WRITE(6,1520)  (X(I,I),I=1,MM)
 1520 FORMAT((5(1X,1PD13.6)))
      IF (IQ.EQ.0) GO TO 389
      DO 9720 I=1,NC
      DO 9720 J=1,NS
      SUM=0.0
      DO 9725 K=1,NS
 9725 SUM=SUM+FBGC(I,K)*GN(K,J)
 9720 X(I,J)=SUM
      DO 9730 I=1,NS
      DO 9730 J=I,NS
      SUM=0.0
      IF (NC.EQ.0) GO TO 9730
      DO 9735 K=1,NC
 9735 SUM=SUM+G(I,K)*X(K,J)
 9730 PRO(I,J)=CQ(I,J)+SUM
      CALL SCOV (NS,SC,W11,CWR,CWI,NS,GM,W21,CR,CI,PRO,BA)
      IF (NC.EQ.0) GO TO 9756
      DO 9755 I=1,NC
      DO 9755 J=1,NS
      W21(I,J)=0.0
      DO 9755 K=1,NS
 9755 W21(I,J)=W21(I,J)+FBGC(I,K)*BA(J,K)
 9756 DO 9760 I=1,NS
      DO 9760 J=1,NS
      SUM=0.0
      IF (NC.EQ.0) GO TO 9760
      DO 9761 K=1,NC
 9761 SUM=SUM+G(I,K)*W21(K,J)
 9760 PRO(I,J)=SUM
      DO 9762 I=1,NS
      DO 9762 J=I,NS
      PRO(I,J)=PRO(I,J)+CQ(I,J)+PRO(J,I)
 9762 PRO(J,I)=PRO(I,J)
      CALL SCOV (NS,SC,W11,CWR,CWI,NS,SC,W11,CWR,CWI,PRO,CQ)
      DO 9770 I=1,NS
      DO 9770 J=I,NS
      GM(I,J)=CQ(I,J)-BA(I,J)-BA(J,I)+GN(I,J)
 9770 GM(J,I)=GM(I,J)
      GOTO 9780
 9360 CALL SCOV (NS,SC,W11,CWR,CWI,NS,SC,W11,CWR,CWI,CQ,GM)
```

```
9780 IF (NC.EQ.0) GO TO 202
     DO 190 I = 1,NS
     DO 190 J = 1,NC
     PRO (I,J) = 0.D0
     DO 191 K = 1,NS
 191 PRO (I,J) = PRO(I,J) +GM(I,K)*FBGC(J,K)
 190 CONTINUE
     DO 200 I = 1,NC
     DO 200 J = 1,NC
     SC(I,J) = 0.D0
     DO 201 K = 1,NS
 201 SC(I,J) =SC(I,J)+FBGC(I,K)*PRO(K,J)
 200 CONTINUE
 202 IF(IREG .EQ. 0) GO TO 9791
     DO 9792 I=1,NS
     DO 9792 J=1,NS
9792 CQ(I,J)=GM(I,J)
     GOTO 503
9791 WRITE(6,220)
 220 FORMAT('0',//,2X,'THE COVARIANCE OF THE ESTIMATE..',//)
     CALL RAPRNT(MH,MH,MH,5,GM,4,'(5(1X,1PD13.6))')
     IF(IR.GT.2) GOTO 503
     DO 67 I = 1,MH
     DO 67 J = 1,MH
  67 CQ(I,J) = GM(I,J)+GM(I,J)
 503 CONTINUE
     WRITE(6,210)
 210 FORMAT('0',//,2X,'THE STATE COVARIANCE MATRIX..',//)
     CALL RAPRNT(MH,MH,MH,5,CQ,4,'(5(1X,1PD13.6))')
     IF (NC.EQ.0) GO TO 232
     WRITE(6,221)
 221 FORMAT('0',//,1X,'THE CONTROL COVARIANCE',//)
     DO 230 I = 1,NC
 230 WRITE(6,231) (SC(I,J),J=1,NC)
 231 FORMAT(1P6D14.6)
 232 DO 240 I = 1,NS
 240 CQ(I,I) = DSQRT(CQ(I,I))
     IF (NC.EQ.0) GO TO 251
     DO 250 I =1,NC
 250 SC(I,I) = DSQRT(SC(I,I))
 251 WRITE(6,262)
 262 FORMAT('0',1X,'STATE RMS RESPONSE',20X,'CONTROL RMS RESPONSE',
    1//)
     DO 270 I=1,NS
     IF (I.LE.NC) WRITE (6,272) CQ (I,I),SC(I,I)
 272 FORMAT(' ',1PD15.7,25X,D15.7)
     IF (I.GT.NC) WRITE (6,272) CQ(I,I)
 270 CONTINUE
 389 IF(ITF3 .EQ. 0) GO TO 440
C
C  FORM COMPENSATOR FROM MEAS TO INPUT AND COMPUTE TF
C
     DO 410 I=1,NS
     DO 410 J=1,NS
     SUM=0.D0
     DO 405 K=1,NO
 405 SUM=SUM+FBGE(I,K)*HO(K,J)
 410 CO(I,J)=ACL(I,J)-SUM
     WRITE(6,9240)
9240 FORMAT('0',//,2X,'COMPENSATOR TRANSFER FUNCTIONS...')
     ITFX=3
     IZERO=0
     CALL TF(NS,NS,NSQ,CQ,AA,NO,FBGE,BM,NC,FBGC,CM,IZERO,D,BB,CC,CP,
    *     WF,WI,CWP,CWI,SC,JCP,RES,D1,D2,DDD,EPS,ITF3,ITFX)
 440 CONTINUE
C
C COMPUTE PSD FUNCTIONS OF THE CONTROLLED SYSTEM
C
     IF(IPSD .EQ. 0) GO TO 450
     IF(IYO .LT. 3) GO TO 444
     CALL PSDCAL(N,NS,RR,X,NC,GW,GY,FBGC,NO,HY,MU,HO,FBGE,NG,
    1 GAM,ACL,BA,WR,WI,D1,D2,JCP,RES,C,RC,BB,CC,1 ,IPSD,INORM)
```

```
      CALL PSDCAL(M,NS,RM,X,NC,GW,GV,FBGC,NO,HY,HY,HO,FBGE,NG,
     1 GAM,ACL,BA,WR,WI,D1,D2,JCP,RES,Q,RC,BB,CC,2    ,IPSD,INORM)
      GO TO 450
  444 CALL PSDCAL(M,NS,RM,X,NC,GW,GV,FBGC,NO,HY,HE,HO,FBGE,NG,
     1 GAM,ACL,BA,WR,WI,D1,D2,JCP,RES,Q,RC,BB,CC,ITO,IPSD,INORM)
  450 IF(ISS .EQ. 0) RETURN
      IF(NC .NE. 0) GO TO 395
      DO 390 I=1,NS
      DO 390 J=1,NS
  390 ACL(I,J) = BA(I,J)
  395 CONTINUE
      CALL MINV(NSQ,ACL,NS,DDD,D1,D2)
      READ(5,7444)    (WR(I),I=1,NG)
      WRITE(6,9771) (WR(I),I=1,NG)
 9771 FORMAT('0',' STEADY DISTURBANCE VECTOR......',/,10(1X,1PD16.6/))
 9765 FORMAT(//////,' STEADY STATE VALUES OF STATE VAR. ARE..........')
      WRITE(6,9765)
      DO 9763 I=1,NS
      WI(I)=0.0
      DO 9763 J=1,NG
 9763 WI(I)=WI(I)+GAM(I,J)*WR(J)
      DO 9764 I=1,NS
      CR(I)=0.0
      DO 9768 J=1,NS
 9768 CR(I)=CR(I)-ACL(I,J)*WI(J)
 9764 WRITE(6,6000) CR(I)
      DO 9766 I=1,NC
      CI(I)=0.0
      DO 9766 J=1,NS
 9766 CI(I)=CI(I)+FBGC(I,J)*CR(J)
      WRITE(6,9767)  (CI(I),I=1,NC)
 9767 FORMAT(///,' STEADY STATE CONTROL IS .........',///,10(1PD15.5/))
      RETURN
      END
      SUBROUTINE CDIV (A,B,C,D,E,F)
      IMPLICIT REAL*8 (A-H,O-Z)
C
C     THIS SUBROUTINE COMPUTES THE COMPLEX DIVISION
C
C              E + F*I = (A + B*I)/(C + D*I)
C
      T=C*C+D*D
      E=(A*C+B*D)/T
      F=(B*C-A*D)/T
C
      RETURN
      END
      SUBROUTINE MAPRNT(NMAX,M,N,L,A,IDIM,FMT)
      REAL*8 A(NMAX,N)
      DIMENSION FMT(IDIM)
      NU=L
      DO 20 NL=1,N,L
      IF (NU.GT.N) NU=N
      DO 10 I=1,M
   10 WRITE(6,FMT)(A(I,J),J=NL,NU)
      WRITE(6,100)
  100 FORMAT(' ')
   20 NU=NU+L
      RETURN
      END
      SUBROUTINE RGAIN(M,NS,NC,NOB,WR,WI,VF,GN,W11,TCB,
     1W21,LT,C,CI,CT,MHS,MT)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION WR(M),WI(M),VF(M,M),GN(NS,NS)
      DIMENSION  W11(NS,NS),TCS(M,M),W21(NS,NS),LT(NS),MT(NS)
      DIMENSION C(NS),CI(NS),CT(NS,NS)
      K = 1
      KP = 1
      KN = 1
      NRZEV = 0
      NCPZEV = 0
   10 IF(K.GT.M) GO TO 200
```

```
C
C CHECK FOR EIGVAL AT OR NEAR J-OMEGA AXIS TO INCLUDE IN E-L EIGSYS
C  TURN FIRST ONE POSITIVE AND SECOND ONE NEGATIVE
C
      EIGVR=DABS(WR(K))
      IF(EIGVR .GE. 1.D-10) GO TO 48
      IF(WI(K)) 40,30,40
   30 NRZEV = NRZEV+1
      IF(NRZEV .GT. 1) GO TO 35
      WR(K) = EIGVR
      GO TO 75
   35 WR(K) = -EIGVR
      WRITE(6,9000)
 9000 FORMAT('0',' EULER-LAGRANGE EQUATIONS HAVE A REAL EIGENVALUE AT',
     *       ' OR NEAR ZERO.'/)
      GO TO 110
   40 NCPZEV = NCPZEV+1
      IF(NCPZEV .GT. 1) GO TO 45
      WR(K) = EIGVR
      WR(K+1) = EIGVR
      GO TO 80
   45 WR(K) =-EIGVR
      WR(K+1) = -EIGVR
      WRITE(6,9010)
 9010 FORMAT('0',' EULER-LAGRANGE EQUATIONS HAVE A COMPLEX PAIR OF ',
     *       'EIGENVALUES AT OR NEAR THE J-OMEGA AXIS.')
      GO TO 120
   48 IF(WR(K)) 100,50,50
   50 IF(WI(K)) 80,75,80
C  ----------EIGENVECTOR FOR REAL EIGENVALUE,POSITIVE
   75 IF(NOB. EQ. 0) GO TO 78
      DO 76 J = 1,N
   76 TCB(J,KP) = VF(J,K)
   78 KP = KP+1
      K=K+1
      GO TO 10
C  ----------EIGENVECTOR FOR COMPLEX EIGENVALUE,POSITIVE REAL PART
   80 IF(NOB .EQ. 0) GO TO 83
      DO 81 J = 1,N
      FR = VF(J,K)
      FI = -VF(J,K+1)
      TCB(J,KP) = FR+FI
   81 TCB(J,KP+1) = FR-FI
   83 KP = KP+2
      K = K+2
      GO TO 10
  100 IF(WI(K)) 120,110,120
C  ----------EIGENVECTOR FOR REAL EIGENVALUE,NEGATIVE REAL PART
  110 C(KN) = WR(K)
      CI(KN) = WI(K)
      IF (NOB.NE. 0) GO TO 96
      KNS = KN+NS
      DO 95 J= 1,N
   95 TCB(J,KNS) = VF(J,K)
   96 KN = KN+1
      K=K+1
      GO TO 10
C  ----------EIGENVECTOR FOR COMPLEX EIGENVALUE,NEGATIVE REAL PART
  120 RR = WR(K)
      RI = WI(K)
      C(KN) = RR
      C(KN+1) = RR
      CI(KN) = RI
      CI(KN+1) = - RI
      IF (NOB.NE. 0) GO TO 122
      KNS = KN+NS
      DO 121 J = 1,N
      FR = VF(J,K)
      FI = -VF(J,K+1)
      TCB(J,KNS) = FR+FI
  121 TCB(J,KNS+1) = FR-FI
  122 KN = KN+2
```

```
      K = K+2
      GO TO 10
  200 CONTINUE
      IF (NOB.NE. 0) GO TO 321
C  FORMATION OF W11
      DO 300 I = 1,NS
      DO 300 J = 1,NS
      W11(I,J) = TCB(I,J+NS)
  300 CT(I,J) = W11(I,J)
C  FORMATION OF W21
      DO 320 I=1,NS
      DO 320 J=1,NS
  320 W21(I,J) = TCB(I+NS,J+NS)
  321 IF (NOB. EQ. 0) GO TO 323
      DO 322 I = 1,NS
      DO 322 J = 1,NS
      W21(I,J) = -TCB(I,J)
  322 W11(I,J) = TCB(I+NS,J)
  323 CONTINUE
C  INVERT W11
      NSQ=NS*NS
      CALL MINV(NSQ,W11,NS,DETC,LT,MT)
C  CALCULATE THE RGAIN MATRIX
      DO 325 IL=1,NS
      DO 325 JL=1,NS
      GN(IL,JL) = 0.D0
      DO 325 KL=1,NS
  325 GN(IL,JL)=GN(IL,JL)+W21(IL,KL)*W11(KL,JL)
      IF (NOB.EQ. 0) RETURN
      DO 5999 I = 1,NS
      DO 5999 J = 1,NS
 5999 CT(I,J) = W11(J,I)
      RETURN
      END
      SUBROUTINE MINV(NSQ,A,N,D,L,M)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(NSQ),L(N),M(N)
      DOUBLE PRECISION A,D,BIGA,HOLD
      NM=N*N
      D=1.0D0
      NK=-N
      DO 60 K=1,N
      NK=NK+N
      L(K)=K
      M(K)=K
      KK=NK+K
      BIGA=A(KK)
      DO 20 J=K,N
      IZ=N*(J-1)
      DO 20 I=K,N
      IJ=IZ+I
   10 IF( DABS(BIGA)- DABS(A(IJ))) 15,20,20
   15 BIGA=A(IJ)
      L(K)=I
      M(K)=J
   20 CONTINUE
C
C        INTERCHANGE ROWS
C
      J=L(K)
      IF(J-K) 35,35,25
   25 KI=K-N
      DO 30 I=1,N
      KI=KI+N
      HOLD=-A(KI)
      JI=KI-K+J
      A(KI)=A(JI)
   30 A(JI) =HOLD
C
C        INTERCHANGE COLUMNS
C
   35 I=M(K)
```

```
      IF(I-K) 45,45,38
   38 JP=N*(I-1)
      DO 40 J=1,N
      JK=NK+J
      JI=JP+J
      HOLD=-A(JK)
      A(JK) =A(JI)
   40 A(JI) =HOLD
C
C        DIVIDE COLUMN BY MINUS PIVOT (VALUE OF PIVOT ELEMENT IS
C        CONTAINED IN BIGA)
C
   45 IF(BIGA) 48,46,48
   46 D=0.0D0
      RETURN
   48 DO 55 I=1,N
      IF(I-K) 50,55,50
   50 IK=NK+I
      A(IK)=A(IK)/(-BIGA)
   55 CONTINUE
C
C        REDUCE MATRIX
C
      DO 65 I=1,N
      IK=NK+I
      HOLD=A(IK)
      IJ=I-N
      DO 65 J=1,N
      IJ=IJ+N
      IF(I-K) 60,65,60
   60 IF(J-K) 62,65,62
   62 KJ=IJ-I+K
      A(IJ) =HOLD*A(KJ)+A(IJ)
   65 CONTINUE
C
C        DIVIDE ROW BY PIVOT
C
      KJ=K-N
      DO 75 J=1,N
      KJ=KJ+N
      IF(J-K) 70,75,70
   70 A(KJ) =A(KJ)/BIGA
   75 CONTINUE
C
C        PRODUCT OF PIVOTS
C
      D=D*BIGA
C
C        REPLACE PIVOT BY RECIPROCAL
C
      A(KK) =(1.0D0)/BIGA
   80 CONTINUE
C
C        FINAL ROW AND COLUMN INTERCHANGE
C
      K=N
  100 K=(K-1)
      IF(K) 150,150,105
  105 I=L(K)
      IF(I-K) 120,120,108
  108 JQ=N*(K-1)
      JR=N*(I-1)
      DO 110 J=1,N
      JK=JQ+J
      HOLD=A(JK)
      JI=JR+J
      A(JK) =-A(JI)
  110 A(JI) =HOLD
  120 J=M(K)
      IF(J-K) 100,100,125
  125 KI=K-N
      DO 130 I=1,N
```

```
      KI=KI+N
      HOLD=A(KI)
      JI=KI-K+J
      A(KI)=-A(JI)
  130 A(JI) =HOLD
      GO TO 100
  150 K=0
      RETURN
      END
      SUBROUTINE SCOV (NL,WL,WLI,VL1,VL2,NR,WR,WRI,VR1,VR2,Q,X)
      REAL*8 VL1(NL),VL2(NL),WL(NL,NL),WLI(NL,NL),X(NL,NR),Q(NL,NR),
     *     VR1(NR),VR2(NR),WR(NR,NR),WRI(NR,NR)
      REAL*8 A,B,C,D,K1,K2,K3,K4
  100 DO 50 I=1,NL
      DO 50 J=1,NR
      X(I,J)=0.
      DO 50 II=1,NL
   50 X(I,J)=X(I,J)+WLI(I,II)*Q(II,J)
      DO 52 I=1,NL
      DO 52 J=1,NR
      Q(I,J)=0.
      DO 51 JJ=1,NR
   51 Q(I,J)=Q(I,J)+X(I,JJ)*WRI(J,JJ)
   52 CONTINUE
      I=1
   13 IF (VL2(I)) 10,11,10
   10 J=1
   14 IF(VR2(J)) 20,21,20
   20 A=VL1(I)+VR1(J)
      B=-2.*VL2(I)*VR2(J)
      C=A**2+VL2(I)**2+VR2(J)**2
      D=C**2-B**2
      K1=A*C/D
      K2=-(VR2(J)*C+VL2(I)*B)/D
      K3=-(VR2(J)*B+VL2(I)*C)/D
      K4=-A*B/D
      I1=I+1
      J1=J+1
      X(I,J)   =+K1*Q(I,J)+K2*Q(I,J1)+K3*Q(I1,J)+K4*Q(I1,J1)
      X(I,J1)  =-K2*Q(I,J)+K1*Q(I,J1)-K4*Q(I1,J)+K3*Q(I1,J1)
      X(I1,J)  =-K3*Q(I,J)-K4*Q(I,J1)+K1*Q(I1,J)+K2*Q(I1,J1)
      X(I1,J1) =+K4*Q(I,J)-K3*Q(I,J1)-K2*Q(I1,J)+K1*Q(I1,J1)
      J=J+2
      GO TO 22
   21 A=VR1(J)+VL1(I)
      B=A**2+VL2(I)**2
      K1=A/B
      K2=VL2(I)/B
      X(I,J)=K1*Q(I,J)-K2*Q(I+1,J)
      X(I+1,J)=K2*Q(I,J)+K1*Q(I+1,J)
      J=J+1
   22 IF(J.LE.NR) GO TO 14
      I=I+2
      GO TO 12
   11 J=1
   15 IF (VR2(J)) 30,31,30
   30 A=VR1(J)+VL1(I)
      B=A**2+VR2(J)**2
      K1=A/B
      K2=VR2(J)/B
      X(I,J)=K1*Q(I,J)-K2*Q(I,J+1)
      X(I,J+1)=K2*Q(I,J)+K1*Q(I,J+1)
      J=J+2
      GO TO 32
   31 X(I,J)=Q(I,J)/(VR1(J)+VL1(I))
      J=J+1
   32 IF (J.LE.NR) GO TO 15
      I=I+1
   12 IF (I.LE.NL) GO TO 13
      DO 40 I=1,NL
      DO 40 J=1,NR
      Q(I,J) =0.
```

```
      DO 40 II=1,NL
   40 Q(I,J) = Q(I,J)  + WL(I,II)*X(II,J)
      DO 42 I=1,NL
      DO 42 J=1,NB
      X(I,J)=0.
      DO 41 JJ=1,NB
   41 X(I,J)=X(I,J)+Q(I,JJ)*WB(J,JJ)
   42 CONTINUE
      RETURN
      END
      SUBROUTINE MODE(WNORM,G,GNORM,NS,N1,N2,ICON)
C
C     WNORM  TRANSFORMATION MATRIX U OR U-INV
C     NS        NO. OF STATE
C     NC        NO.  OF INPUTS OR OUTPUTS
C     ICON   CONTROL FLAG TO INDICATE WHICH TRANSFORMATION
C                0 =   MODAL G
C                1 =   MODAL GAMMA
C                2 =   MODAL H
C                3 =   MODAL C
C                4 =   MODAL K
C                5 =   CONTROL EIGENVECTOR MATRIX
C                6 =   MEASUREMENT EIGENVECTOR MATRIX
C
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION WNORM(NS,NS),G(N1,N2),GNORM(N1,N2)
 6500 FORMAT('0',//,2X,'MODAL CONTROL DISTRIBUTION MATRIX..',//)
 6501 FORMAT('0',//,2X,'MODAL PROCESS NOISE DISTRIBUTION MATRIX..',//)
 6570 FORMAT('0',//,2X,'MODAL MEASUREMENT SCALING MATRIX',//)
 6580 FORMAT(///,' ','THE MODAL CONTROL GAINS ARE:',//)
 6584 FORMAT('0',/,'  CONTROL EIGENVECTOR MATRIX..',//)
 6586 FORMAT('0',/,'  MEASUREMENT EIGENVECTOR MATRIX..',//)
 6000 FORMAT(' ',(2X,1P6D14.6))
 6590 FORMAT('0','MODAL FILTER STEADY STATE GAINS......',//)
      DO 50 I=1,N1
      DO 50 J=1,N2
   50 GNORM(I,J)=0.
      IPOINT=ICON+1
      GO TO (75,75,250,250,75,250,250),IPOINT
   75 DO 100 J=1,N2
      DO 100 I=1,NS
      DO 100 K=1,NS
  100 GNORM(I,J)=GNORM(I,J)+WNORM(I,K)*G(K,J)
      GO TO (105,150,250,250,160),IPOINT
  105 WRITE(6,6500)
  110 DO 120 I=1,NS
  120 WRITE(6,6000) (GNORM(I,J),J=1,N2)
      RETURN
  150 WRITE(6,6501)
      GO TO 110
  160 WRITE(6,6590)
      GO TO 110
  250 DO 260 J=1,NS
      DO 260 I=1,N1
      DO 260 K=1,NS
  260 GNORM(I,J) = GNORM(I,J) + G(I,K)*WNORM(K,J)
      GO TO (261,261,261,262,261,263,264),IPOINT
  261 WRITE(6,6570)
      GO TO 270
  262 WRITE(6,6580)
      GO TO 270
  263 WRITE(6,6584)
      GO TO 270
  264 WRITE(6,6586)
  270 DO 275 I=1,N1
  275 WRITE(6,6000) (GNORM(I,J),J=1,NS)
      RETURN
      END
      SUBROUTINE CNORM(WZ,WY,VEC,NS,IWRITE,NSQ,DDD,D1,D2,WNORM,WNORMI,
     8   HO,CN,N1,N2)
C
C          WZ(I)        REAL PART OF I-TH EIGENVALUE
```

```
C
C         WI(I)       COMPLEX PART OF I-TH EIGENVALUE
C
C         VEC         MATRIX OF RIGHT EIGENVECTORS STORED IN REAL FORM
C                       FROM HQR2
C         NS          NO. OF STATES
C
C         IWRITE      FLAG TO CONTROL FORMATS FOR DIFFERENT EIGENSYSTEMS
C
C         WNORM       NORMALIZED MATRIX U OF RIGHT EIGENVECTORS STORED
C                       BY COLUMNS IN REAL FORM
C         WNORMI      U-INVERSE 2*CONGUGATE OF LEFT EIGENVECTORS
C                       STORED BY ROW IN REAL FORM
C         NSQ,DDD,D1,D2 - ARGUMENTS PASSED TO MINV
C
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 FIELD,COMMA,SEMCOL,RIGHT,FMT
      DIMENSION WZ(NS),WI(NS),VEC(NS,NS),WNORM(NS,NS)
     1     ,WNORMI(NS,NS),STORE(6),D1(NS),D2(NS),FMT(14),HO(N1,N2),
     *     CM(N1,N2)
      DATA FIELD/5HE12.5/,COMMA/5H,',', /,SEMCOL/5H,':',/
     1    ,RIGHT/1H)/,FMT/6H(1X,1P,13*1H /,SEMEND/4H,':'/
 9030 FORMAT('0','OPEN LOOP EIGENVALUES..')
 9040 FORMAT('0','CLOSED LOOP OPTIMAL REGULATOR EIGENVALUES..')
 9050 FORMAT('0','CLOSED LOOP SUBOPTIMAL REGULATOR EIGENVALUES..')
 9060 FORMAT('0','CLOSED LOOP OPTIMAL ESTIMATOR EIGENVALUES..')
 9070 FORMAT('0','CLOSED LOOP SUBOPTIMAL ESTIMATOR EIGENVALUES..')
 9080 FORMAT(//'0','RIGHT EIGENVECTOR MATRIX..'//)
 9090 FORMAT(//'0','OPEN LOOP LEFT EIGENVECTOR MATRIX..')
 9100 FORMAT(//'0','CLOSED LOOP OPT. REG. LEFT EIGENVECTOR MATRIX..')
 9110 FORMAT(//'0','CLOSED LOOP SUBOPT. REG. LEFT EIGENVECTOR MATRIX..'
 9120 FORMAT(//'0','CLOSED LOOP OPT. FILTER LEFT EIGENVECTOR MATRIX..')
 9130 FORMAT(//'0','CLOSED LOOP SUBOPT. FILTER LEFT EIGENVECTOR MATRIX.
     1')
C  11 FORMAT (46X,'(',F10.7,')+J(',F10.7,')')
C
C NORMALIZE COMPLEX EIGENVECTORS BY LARGEST ELEMENT
C
C
      KK=0
      LK=0
      LC=0
      DO 999 K =1,NS
      IF(KK.EQ.1) GOTO 991
      IF (DABS(WI(K)).LT.1.D-10)  GO TO 999
      LC=LC+1
      EMAX = 0.DO
      DO 997 I = 1,NS
      CMOD=VEC(I,K )**2+VEC(I,K+1)**2
      IF (CMOD-EMAX) 997,990,990
  990 EMAX = CMOD
      M=I
  997 CONTINUE
      VMR = VEC(M,K)
      VMI = VEC(M,K+1)
      DO 980 I=1,NS
      VR = VEC(I,K)
      VI = VEC(I,K+1)
      VECRN=(VR*VMR+VI*VMI)/EMAX
      VECIN=(-VR*VMI+VI*VMR)/EMAX
      WNORM(I,K)=VECRN
      WNORM(I,K+1)=VECIN
  980 CONTINUE
      KK=1
      GOTO 999
 991  KK=0
  999 CONTINUE
C
C NORMALIZE REAL EIGENVECTORS BY THE TOTAL LENGTH
C
      DO 1000 K=1,NS
      IF(DABS(WI(K)).GE.1.D-10) GOTO 1000
```

```
  998 LR=LR+1
      REMOD = 0.D0
      DO 996 I=1,NS
  996 REMOD=VEC(I,K )**2+REMOD
      RMOD=DSQRT(REMOD)
      DO 995 I=1,NS
      RVEC=VEC(I,K)/RMOD
      WNORM(I,K)=RVEC
  995 CONTINUE
 1000 CONTINUE
C
      GO TO (520,530,540,545,550),IWRITE
  520 WRITE(6,9030)
      GO TO 560
  530 WRITE(6,9040)
      GO TO 560
  540 WRITE(6,9050)
      GO TO 560
  545 WRITE(6,9060)
      GO TO 560
  550 WRITE(6,9070)
  560 KK=0
      NPRTW=0
      NFMTW=1
      DO 568 I=1,NS
      IF(KK .EQ. 1) GO TO 567
      IF(DABS(WY(I)) .GT. 1.D-10)  KK=1
C
C PRINT OUT NO MORE THAN 6 WORDS, NOT SEPARATING COMPLEX EIGVAL
C
      IF(NPRTW .LT. 5 .OR.(NPRTW .EQ. 5 .AND. KK .EQ. 0)) GO TO 561
      FMT(NFMTW+1) = RIGHT
      WRITE(6,FMT)  (STORE(J),J=1,NPRTW)
      NPRTW=0
      NFMTW=1
  561 NPRTW=NPRTW+1
      NFMTW=NFMTW+1
      IF(KK .EQ. 1) GO TO 562
      STORE(NPRTW) = WZ(I)
      FMT(NFMTW) = FIELD
      NFMTW = NFMTW+1
      FMT(NFMTW) = SEMCOL
      GO TO 568
  562 STORE(NPRTW) = WZ(I)
      FMT(NFMTW) = FIELD
      FMT(NFMTW+1) = COMMA
      STORE(NPRTW+1) = WY(I)
      FMT(NFMTW+2) = FIELD
      FMT(NFMTW+3) = SEMCOL
      NFMTW = NFMTW + 3
      NPRTW = NPRTW + 1
      GO TO 568
  567 KK=0
  568 CONTINUE
      FMT(NFMTW)=SEMEND
      FMT(NFMTW+1) = RIGHT
      WRITE(6,FMT)  (STORE(J),J=1,NPRTW)
      WRITE(6,9080)
C     CALL MAPRNT(NS,NS,NS,6,WNORM,4,'(6(1X,1PD13.6))')
      CALL MAPRNT(NS,NS,NS,6,WNORM,4,'(6(1PG12.6))')
      GO TO (578,570,570,575,575),IWRITE
  570 CALL MODE(WNORM,HO,CM,NS,N1,N2,5)
      GO TO 578
  575 CALL MODE(WNORM,HO,CM,NS,N1,N2,6)
  578 GO TO   (580,590,600,610,620),IWRITE
  580 WRITE(6,9090)
      GO TO 630
  590 WRITE(6,9100)
      GO TO 630
  600 WRITE(6,9110)
      GO TO 630
  610 WRITE(6,9120)
```

```
      GO TO 630
  620 WRITE(6,9130)
C SAVE U-INV OPEN LOOP IN WNORMI
  630 IF(IWRITE .GT. 1) GO TO 1005
      DO 510 I=1,NS
      DO 510 J=1,NS
  510 WNORMI(I,J)=WNORM(I,J)
      CALL MINV(NSQ,WNORMI,NS,DDD,D1,D2)
      CALL RAPRNT(NS,NS,NS,6,WNORMI,4,'(6(1X,1PD13.6))')
      RETURN
 1005 CALL MINV(NSQ,WNORM,NS,DDD,D1,D2)
      CALL RAPRNT(NS,NS,NS,6,WNORM,4,'(6(1X,1PD13.6))')
      RETURN
      END
      SUBROUTINE TF(N,NM,NSQ,A,AA,M,B,BM,L,C,CM,IFDFW,D,BB,CC,CP,
     *     EVR,EVI,PR,PI,SC,JCP,RES,D1,D2,DDD,EPS,ITP,ITFX)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION A(N,N),AA(N,N),B(N,M),BM(N,M),C(L,N),CM(L,N),D(L,M),
     *     BB(N),CC(N),CP(N),EVR(N),EVI(N),PR(N),PI(N),SC(N,N),JCP(N),
     *     RES(N),D1(N),D2(N)
C  SAVE COMPUTATIOON ON OL AND CL SYS WITH MODAL WORK DONE IN OPTSYS
      IF(ITFX .EQ. 1) GO TO 50
      IF(ITFX .EQ. 2) GO TO 5
      CALL POLES(N,NM,A,AA,M,B,L,C,PR,PI,D1,D2,JCP,SC)
C  COMPUTE MODAL MATRICES FOR RESIDUES
    5 DO 10 I=1,N
      DO 10 J=1,N
   10 AA(I,J) = SC(I,J)
   15 DO 20 I=1,L
      DO 20 J=1,N
      CM(I,J) = 0.D0
      DO 20 K=1,N
   20 CM(I,J) = CM(I,J) + C(I,K)*AA(K,J)
      CALL MINV(NSQ,AA,N,DDD,D1,D2)
      DO 30 I=1,N
      DO 30 J=1,M
      BM(I,J)= 0.D0
      DO 30 K=1,N
   30 BM(I,J) =BM(I,J) + AA(I,K)*B(K,J)
   50 CONTINUE
      DO 100 I=1,M
      DO 100 J=1,L
      IF(ITP .NE. 3)
     *CALL ZEROS(I,J,IFDFW,N,NM,A,AA,M,B,L,C,D,BB,CC,CP,EVR,EVI,D1,D2,
     *     EPS)
      IF(ITP .NE. 2) CALL RESID(I,J,N,JCP,M,BM,L,CM,PR,PI,RES,BB,CC,1)
  100 CONTINUE
      RETURN
      END
      SUBROUTINE CHECK(EPS,NC,NS,NO)
      DOUBLE PRECISION EPS
      COMMON /PROG/ IOL,INQ,IQ,IR,ISS,IM,ITF1,ITF2,ITF3,IFDFW,IE,IDSTAB
     *     IDEBUG,ISET,IREG,IPSD,IFU,INORM
C  SET MODAL ANALYSIS WHEN OL EIGEN SYS OR OL TF REQUESTED
      IF(IM .EQ. 1 .AND. IOL .EQ. 0) IOL=1
      IF(IOL .EQ. 3 .OR. ITF1 .NE. 0) IM=1
C  CHECK TO SEE IF H MATRIX INPUT
      IF(NO .NE. 0 .OR. IOL .GE. 2) GO TO 25
      WRITE(6,8000)
 8000 FORMAT(//' H - MATRIX MUST BE INPUT, I.E. NOB .GT. 0')
      STOP
   25 CONTINUE
C
C  TRANSFER FUNCTION CHECKS
C
      IF(IE .EQ. 0) IE=6
      EPS=10.**(-IE)
C  OPEN LOOP TF
      IF(ITF1 .EQ. 0 .OR. NC .NE . 0) GO TO 50
      WRITE(6,9000)
 9000 FORMAT(//' INPUT(G) MATRIX MUST BE REQUESTED(I.E. NC .NE. 0) ',
     *     'TO COMPUTE OPEN LOOP T. F. ')
```

```
      STOP
C   COMPENSATOR TF
   50 IF(ITF3 .EQ. 0) GO TO 100
      IF(IREG .EQ. 0 .AND. (NC .NE. 0 .AND. NG .NE. 0)) GO TO 100
      WRITE(6,9100)
 9100 FORMAT(//' REGULATOR AND FILTER SYNTHESIS MUST BE REQUESTED IN '
     *      ,'THE SAME RUN TO COMPUTE COMPENSATOR T. F.')
      STOP
  100 CONTINUE
C NOISE TF
      IF(ITF2 .EQ. 0) GO TO 150
      IF(NG.NE. 0 .AND. NC .NE. 0) GO TO 150
      WRITE(6,9200)
 9200 FORMAT(//' NOISE T. F. CALCULATED ONLY WHEN REGULATOR DESIGNED '
     *      ,'AND GAMMA INPUT, I. E. NG .NE. 0')
      STOP
C
C   DESTABILIZATION RESTRICTIONS
  150 IF(IDSTAB .EQ. 0) GO TO 200
      IF(NC .EQ. 0) GO TO 200
      IF(NG .NE. 0) IREG=1
      WRITE(6,9300)
 9300 FORMAT(//' DESTABILIZATION OPTION DESIGNED FOR A REGULATOR OR ',
     *      'FILTER BUT NOT BOTH SIMULTANEOUSLY')
      IF(IREG .EQ. 1) GO TO 200
      STOP
  200 CONTINUE
C
C PSD INPUT
C
      IF(IPSD .EQ. 0) GO TO 300
      IF(IPSD .LT. 0 .OR. IPSD .GT. 3) GO TO 250
      IF(IYU .LT. 0 .OR. IYU .GT. 2) GO TO 250
      IF(INORM .LT. 0 .OR. INORM .GT. NG+ND) GO TO 250
      GO TO 275
  250 WRITE(6,9400)
 9400 FORMAT(' ********* INCONSISTENT PSD INPUT FLAGS *********')
      STOP
  275 IF(IREG .EQ. 0 .AND. NC .NE. 0) GO TO 300
      WRITE(6,9500)
 9500 FORMAT(' ********BOTH A REGULATOR AND FILTER MUST BE RESIDENT ',
     1 'TO COMPUTE THE PSD OF A CONTROLLED SYSTEM!')
      STOP
  300 CONTINUE
      RETURN
      END
      SUBROUTINE POLES(N,NM,A,AA,B,B,L,C,EVR,EVI,D1,D2,JCF,SC)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION A(N,N),AA(N,N),B(N,M),C(L,N),EVR(N),EVI(N),D1(N),D2(N),
     *      JCF(N),SC(N,N)
      DO 1 I=1,N
      DO 1 J=1,N
    1 AA(I,J)=A(I,J)
C*****************************
      CALL BALANC(NM,N,AA,LOW,IHIGH,D1)
      CALL ORTHES(NM,N,LOW,IHIGH,AA,D2)
      CALL ORTRAN(NM,N,LOW,IHIGH,AA,D2,SC)
      CALL HQR2(NM,N,LOW,IHIGH,AA,EVR,EVI,SC,IERR)
      IF(IERR .NE. 0) GO TO 110
      CALL BALBAK(NM,N,LOW,IHIGH,D1,N,SC)
C*****************************
      WRITE(6,101)
  101 FORMAT(///,28H TF DENOMINATOR EIGENVALUES:)
      DO 2 I=1,N
    2 WRITE(6,102) EVR(I),EVI(I)
  102 FORMAT(/,2X,3H  (,F13.6,4H) +J(,F13.6,1H))
      RETURN
  110 WRITE(6,9000)
 9000 FORMAT(' FAILURE IN HQR2, CALCULATING POLES')
  100 RETURN
      END
      SUBROUTINE ZEROS(K1,K2,IFDFW,N,NM,A,AA,M,B,L,C,D,BB,CC,CP,EVR,EVI
```

```
*     D1,D2,EPS)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(N,N),AA(N,N),B(N,M),C(L,N),D(L,M),BB(N),CC(N),CP(N),
     *     EVR(N),EVI(N),D1(N),D2(N)
      DOUBLE PRECISION SCL,DABS
      DO 1 I=1,N
      BB(I)  = B(I,K1)
      CC(I)  = C(K2,I)
      DO 1 J=1,N
    1 AA(I,J)=A(I,J)
      WRITE(6,101) K1,K2
  101 FORMAT(///,17H TF FOR INPUT NO.,I3,15H AND OUTPUT NO.,I3,':')
      IF(IFDFW .EQ. 0) GO TO 2
      H= D(K2,K1)
      IF(DABS(H).LE.EPS) GOTO 2
      JJ=N
      GO TO 5
    2 NN=N-1
      DO 3 I=1,NN
      H=SCL(N,BB,CC)
      CALL CCOMP(N,NM,AA,CC,CP)
      IF(DABS(H).GT.EPS) GO TO 4
    3 CONTINUE
      H=SCL(N,BB,CC)
      WRITE(6,102) H
  102 FORMAT(//,5X,26HNO FINITE ZEROS.  TF GAIN=,F13.6)
      GO TO 100
    4 JJ=N-I
    5 WRITE(6,103) JJ,H
  103 FORMAT(/,3X,20HORDER OF NUMERATOR =,I3,9X,8HTF GAIN=,F13.6)
      CALL ACOMP(N,NM,AA,BB,CC,H)
C[illegible]
      CALL BALANC(NM,N,AA,LOW,IHIGH,D1)
      CALL ORTHES(NM,N,LOW,IHIGH,AA,D2)
      CALL HQR(NM,N,LOW,IHIGH,AA,EVR,EVI,IERR)
      IF(IERR .NE. 0) GO TO 110
C[illegible]
      WRITE(6,104)
  104 FORMAT(/,3X,57HNUMERATOR EIGENVALUES (INCLUDING EXTRANEOUS ZERO V
     1LUES):)
      DO 6 I=1,N
    6 WRITE(6,105) EVR(I),EVI(I)
  105 FORMAT(/,4X,'(',F13.6,')+J(',F13.6,')')
  100 RETURN
  110 WRITE(6,9000)
 9000 FORMAT(' FAILURE IN HQR CALCULATING TRANSFER FUNCTION ZEROES')
      RETURN
      END
      SUBROUTINE ACOMP(N,NM,A,B,C,H)
      REAL*8 A,B,C,H
      DIMENSION A(NM,N),B(N),C(N)
      DO 1 I=1,N
      DO 1 J=1,N
    1 A(I,J)=A(I,J)-B(I)*C(J)/H
      RETURN
      END
      SUBROUTINE CCOMP(N,NM,A,C,CC)
      REAL*8 A,C,CC
      DIMENSION A(NM,N),C(N),CC(N)
      DO 1 I=1,N
      CC(I)=0.
      DO 1 J=1,N
    1 CC(I)=CC(I)+C(J)*A(J,I)
      DO 2 I=1,N
    2 C(I)=CC(I)
      RETURN
      END
      FUNCTION SCL(N,B,C)
      REAL*8 B,C,SCL
      DIMENSION B(N),C(N)
      SCL=0.
      DO 1 I=1,N
```

```
    1 SCL=SCL+C(I)*B(I)
      RETURN
      END
      SUBROUTINE RESID(K1,K2,N,JCP,M,BM,L,CM,PR,PI,RES,BB,CC,IPT)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION JCP(N),BM(N,M),CM(L,N),PR(N),PI(N),RES(N),BB(N),CC(N),
     * PRT(4)
      DATA SN/8H*SIN(B*T/,R1/8H        */,R2/8HEXP(A*T)/
      DATA ZERO/0.D0/,T1/8H*T**/, BLANK/8H        /,CS/8H*COS(B*T/
      DATA ED/1H)/
C  TEMPORARY MOD TILL JCP IS CALCULATED
      DO 5 I=1,N
    5 JCP(I)=0
C  TEMPORARY MOD
      IF(IPT .EQ. 1) WRITE(6,9000)
 9000 FORMAT(//,3X,'RESIDUES AT THE POLES:'/,T16,'P O L E S',T41,
     *  'R E S I D U E S',/,T9,'REAL(A)',T26,'IMAG(B)')
      DO 10 I=1,N
      BB(I) = BM(I,K1)
   10 CC(I)= CM(K2,I)
C
C  LOCP THROUGH THE POLES
C
      I=0
  100 I=I+1
      IF(I .GT. N) GO TO 500
      IF(JCP(I) .EQ. 1) GO TO 300
      IF(DABS(PI(I)) .LT. 1.D-10) GO TO 200
C COMPUTE SIMPLE COMPLEX POLE RESIDUES AND PRINT BOTH
      RES(I) = CC(I)*BB(I) + CC(I+1)*BB(I+1)
      RES(I+1)=CC(I)*BB(I+1) - CC(I+1)*BB(I)
      IF(IPT .EQ. 0) GO TO 110
      PRT(1)= BLANK
      PRT(2) = R2
      IF(PI(I) .EQ. 0.D0) PRT(2) = BLANK
      PRT(3) = CS
      PRT(4) = ED
      WRITE(6,9020) PR(I),PI(I),RES(I),(PRT(J),J=1,4)
      I=I+1
      PRT(3)=SN
      WRITE(6,9020) PR(I),PI(I),RES(I),(PRT(J),J=1,4)
      GO TO 100
  110 I = I + 1
      GO TO 100
  200 CONTINUE
C  COMPUTE SIMPLE PEAL POLE RESIDUE
      RES(I) = CC(I)*BB(I)
      IF(IPT .EQ. 0) GO TO 100
      PRT(1)= R1
      PRT(2)= R2
      PRT(3)= BLANK
      PRT(4)= BLANK
      WRITE(6,9020) PR(I),PI(I),RES(I),(PRT(J),J=1,4)
      GO TO 100
C  LOOK AHEAD TO DETERMINE SIZE OF THE JORDAN BLOCK
  300 K=1
      KT=N-I
      DO 310 J=I,KT
      IF(JCP(J) .EQ. 0) GO TO 320
  310 K=K+1
  320 CONTINUE
      IF(DABS(PI(I)) .LT. 1.D-10) GO TO 400
C  COMPUTE REPEATED COMPLEX POLE AND PRINT OUT ALL FOUR
      K=1
      RES(I)=CC(I)*BB(I)+CC(I+1)*BB(I+1)+CC(I+2)*BB(I+2)+CC(I+3)*BB(I+3
      RES(I+1)=CC(I)*BB(I+1)-CC(I+1)*BB(I)+CC(I+2)*BB(I+3)-CC(I+3)*
     X     BB(I+2)
      RES(I+2)=CC(I)*BB(I+3)+CC(I+1)*BB(I+2)
      RES(I+3)=CC(I)*BB(I+3)-CC(I+1)*BB(I+2)
      IF(IPT .EQ. 0) GO TO 340
      PRT(1)=R1
      PRT(2)=R2
```

```
      IF(DABS(PR(I)) .GT. 1.D-10) GO TO 330
      PRT(1)= BLANK
      PRT(2)= BLANK
  330 PRT(3)=CS
      PRT(4)=ED
      WRITE(6,9020) PR(I),PI(I),RES(I),(PRT(J),J=1,4)
 9020 FORMAT(/,4X,'(',F13.6,')+J(',F13.6,') ',4X,'(',F13.6,')',3A8,A1)
 9030 FORMAT(/,4X,'(',F13.6,')+J(',F13.6,') ',4X,'(',F13.6,')',A4,I2,2X,
     *       2A8,A1)
      PRT(3)=SW
      I=I+1
      WRITE(6,9020) PR(I),PI(I),RES(I),(PRT(J),J=1,4)
      PRT(1)=T1
      PRT(2)=R2
      IF(DABS(PR(I)) .LT. 1.D-10) PRT(2)=BLANK
      PRT(3)= CS
      I=I+1
      WRITE(6,9030)  PR(I),PI(I),RES(I),PRT(1),K,(PRT(J),J=2,4)
      PRT(3)=SW
      I=I+1
      WRITE(6,9030)  PR(I),PI(I),RES(I),PRT(1),K,(PRT(J),J=2,4)
      GO TO 100
  340 I = I + 3
      GO TO 100
C COMPUTE REPEATED REAL POLE RESIDUE AND PRINT OUT ALL K OF THEM
  400 CONTINUE
      KT=I+K-1
      NN=0
      DO 420 J=I,KT
      NN= NN+1
      RES(J)=ZERO
      DO 410 JJ=J,KT
  410 RES(J)=RES(J)+BB(JJ)*CC(JJ-NN+1)
  420 CONTINUE
      IF(IPT .EQ. 0) GO TO 440
      NN=0
      PRT(1)=T1
      PRT(2)=R2
      PRT(3)=BLANK
      PRT(4)=BLANK
      DO 430 J=I,KT
      WRITE(6,9030) PR(J),PI(J),RES(J),PRT(1),NN,(PRT(JJ),JJ=2,4)
  430 NN=NN+1
      GO TO 100
  440 I = KT
      GO TO 100
  500 CONTINUE
      RETURN
      END
C
C     ------------------------------------------------------------------
C
      SUBROUTINE BALANC(NM,N,A,LOW,IGH,SCALE)
C
      INTEGER I,J,K,L,M,N,JJ,NM,IGH,LOW,IEXC
      REAL*8 A(NM,N),SCALE(N)
      REAL*8 C,F,G,R,S,B2,RADIX
      REAL*8 DABS
      LOGICAL NOCONV
      DATA RADIX/Z4210000000000000/
C
      B2 = RADIX * RADIX
      K = 1
      L = N
      GO TO 100
C     :::::::::: IN-LINE PROCEDURE FOR ROW AND
C                COLUMN EXCHANGE ::::::::::
   20 SCALE(M) = J
      IF (J .EQ. M) GO TO 50
C
      DO 30 I = 1, L
         F = A(I,J)
```

```
         A(I,J) = A(I,M)
         A(I,M) = F
   30 CONTINUE
C
      DO 40 I = K, N
         F = A(J,I)
         A(J,I) = A(M,I)
         A(M,I) = F
   40 CONTINUE
C
   50 GO TO (80,130), IEXC
C     :::::::::: SEARCH FOR ROWS ISOLATING AN EIGENVALUE
C                AND PUSH THEM DOWN ::::::::::
   80 IF (L .EQ. 1) GO TO 280
      L = L - 1
C     :::::::::: FOR J=L STEP -1 UNTIL 1 DO -- ::::::::::
  100 DO 120 JJ = 1, L
         J = L + 1 - JJ
C
         DO 110 I = 1, L
            IF (I .EQ. J) GO TO 110
            IF (A(J,I) .NE. 0.0D0) GO TO 120
  110    CONTINUE
C
         M = L
         IEXC = 1
         GO TO 20
  120 CONTINUE
C
      GO TO 140
C     :::::::::: SEARCH FOR COLUMNS ISOLATING AN EIGENVALUE
C                AND PUSH THEM LEFT ::::::::::
  130 K = K + 1
C
  140 DO 170 J = K, L
C
         DO 150 I = K, L
            IF (I .EQ. J) GO TO 150
            IF (A(I,J) .NE. 0.0D0) GO TO 170
  150    CONTINUE
C
         M = K
         IEXC = 2
         GO TO 20
  170 CONTINUE
C     :::::::::: NOW BALANCE THE SUBMATRIX IN ROWS K TO L ::::::::::
      DO 180 I = K, L
  180 SCALE(I) = 1.0D0
C     :::::::::: ITERATIVE LOOP FOR NORM REDUCTION ::::::::::
  190 NOCONV = .FALSE.
C
      DO 270 I = K, L
         C = 0.0D0
         R = 0.0D0
C
         DO 200 J = K, L
            IF (J .EQ. I) GO TO 200
            C = C + DABS(A(J,I))
            R = R + DABS(A(I,J))
  200    CONTINUE
C     :::::::::: GUARD AGAINST ZERO C OR R DUE TO UNDERFLOW ::::::::::
         IF (C .EQ. 0.0D0 .OR. R .EQ. 0.0D0) GO TO 270
         G = R / RADIX
         F = 1.0D0
         S = C + R
  210    IF (C .GE. G) GO TO 220
         F = F * RADIX
         C = C * B2
         GO TO 210
  220    G = R * RADIX
  230    IF (C .LT. G) GO TO 240
         F = F / RADIX
```

```
      C = C / B2
      GO TO 230
C     :::::::::: NOW BALANCE ::::::::::
  240 IF ((C + R) / F .GE. 0.95D0 * S) GO TO 270
      G = 1.0D0 / F
      SCALE(I) = SCALE(I) * F
      NOCONV = .TRUE.
C
      DO 250 J = K, N
  250 A(I,J) = A(I,J) * G
C
      DO 260 J = 1, L
  260 A(J,I) = A(J,I) * F
C
  270 CONTINUE
C
      IF (NOCONV) GO TO 190
C
  280 LOW = K
      IGH = L
      RETURN
C     :::::::::: LAST CARD OF BALANC ::::::::::
      END
C
C     ------------------------------------------------------------------
C
      SUBROUTINE ORTHES(NM,N,LOW,IGH,A,ORT)
C
      INTEGER I,J,M,N,II,JJ,LA,MP,NM,IGH,KP1,LOW
      REAL*8 A(NM,N),ORT(IGH)
      REAL*8 F,G,H,SCALE
      REAL*8 DSQRT,DABS,DSIGN
C
      LA = IGH - 1
      KP1 = LOW + 1
      IF (LA .LT. KP1) GO TO 200
C
      DO 180 M = KP1, LA
         H = 0.0D0
         ORT(M) = 0.0D0
         SCALE = 0.0D0
C     :::::::::: SCALE COLUMN (ALGOL TOL THEN NOT NEEDED) ::::::::::
         DO 90 I = M, IGH
   90    SCALE = SCALE + DABS(A(I,M-1))
C
         IF (SCALE .EQ. 0.0D0) GO TO 180
         MP = M + IGH
C     :::::::::: FOR I=IGH STEP -1 UNTIL M DO -- ::::::::::
         DO 100 II = M, IGH
            I = MP - II
            ORT(I) = A(I,M-1) / SCALE
            H = H + ORT(I) * ORT(I)
  100    CONTINUE
C
         G = -DSIGN(DSQRT(H),ORT(M))
         H = H - ORT(M) * G
         ORT(M) = ORT(M) - G
C     :::::::::: FORM (I-(U*UT)/H) * A ::::::::::
         DO 130 J = M, N
            F = 0.0D0
C     :::::::::: FOR I=IGH STEP -1 UNTIL M DO -- ::::::::::
            DO 110 II = M, IGH
               I = MP - II
               F = F + ORT(I) * A(I,J)
  110       CONTINUE
C
            F = F / H
C
            DO 120 I = M, IGH
  120       A(I,J) = A(I,J) - F * ORT(I)
C
  130    CONTINUE
```

```
C     :::::::::: FORM (I-(U*UT)/H)*A*(I-(U*UT)/H) ::::::::::
         DO 160 I = 1, IGH
            F = 0.0D0
C     :::::::::: FOR J=IGH STEP -1 UNTIL M DO -- ::::::::::
            DO 140 JJ = M, IGH
               J = MP - JJ
               F = F + ORT(J) * A(I,J)
  140       CONTINUE
C
            F = F / H
C
            DO 150 J = M, IGH
  150       A(I,J) = A(I,J) - F * ORT(J)
C
  160    CONTINUE
C
         ORT(M) = SCALE * ORT(M)
         A(M,M-1) = SCALE * G
  180 CONTINUE
C
  200 RETURN
C     :::::::::: LAST CARD OF ORTHES ::::::::::
      END
C
C     ------------------------------------------------------------
C
      SUBROUTINE ORTRAN(NM,N,LOW,IGH,A,ORT,Z)
C
      INTEGER I,J,N,KL,MM,MP,NM,IGH,LOW,MP1
      REAL*8 A(NM,IGH),ORT(IGH),Z(NM,N)
      REAL*8 G
C
C     :::::::::: INITIALIZE Z TO IDENTITY MATRIX ::::::::::
      DO 80 I = 1, N
C
         DO 60 J = 1, N
   60    Z(I,J) = 0.0D0
C
         Z(I,I) = 1.0D0
   80 CONTINUE
C
      KL = IGH - LOW - 1
      IF (KL .LT. 1) GO TO 200
C     :::::::::: FOR MP=IGH-1 STEP -1 UNTIL LOW+1 DO -- ::::::::::
      DO 140 MM = 1, KL
         MP = IGH - MM
         IF (A(MP,MP-1) .EQ. 0.0D0) GO TO 140
         MP1 = MP + 1
C
         DO 100 I = MP1, IGH
  100    ORT(I) = A(I,MP-1)
C
         DO 130 J = MP, IGH
            G = 0.0D0
C
            DO 110 I = MP, IGH
  110       G = G + ORT(I) * Z(I,J)
C     :::::::::: DIVISOR BELOW IS NEGATIVE OF H FORMED IN ORTHES.
C                DOUBLE DIVISION AVOIDS POSSIBLE UNDERFLOW ::::::::::
            G = (G / ORT(MP)) / A(MP,MP-1)
C
            DO 120 I = MP, IGH
  120       Z(I,J) = Z(I,J) + G * ORT(I)
C
  130    CONTINUE
C
  140 CONTINUE
C
  200 RETURN
C     :::::::::: LAST CARD OF ORTRAN ::::::::::
      END
C
```

```
C     ------------------------------------------------------------------
C
      SUBROUTINE HQR2(NM,N,LOW,IGH,H,WR,WI,Z,IERR)
C
      INTEGER I,J,K,L,M,N,EN,II,JJ,LL,MM,NA,NM,NN,
     X        IGH,ITS,LOW,MP2,ENM2,IERR
      REAL*8 H(NM,N),WR(N),WI(N),Z(NM,N)
      REAL*8 P,Q,R,S,T,W,X,Y,RA,SA,VI,VR,ZZ,NORM,MACHEP
      REAL*8 DSQRT,DABS,DSIGN
      INTEGER MINO
      LOGICAL NOTLAS
      COMPLEX*16 Z3
      COMPLEX*16 DCMPLX
      REAL*8 DREAL,DIMAG
C     :::::::::: STATEMENT FUNCTIONS ENABLE EXTRACTION OF REAL AND
C                IMAGINARY PARTS OF DOUBLE PRECISION COMPLEX NUMBERS ::::::::::
      DREAL(Z3) = Z3
      DIMAG(Z3) = (0.0D0,-1.0D0) * Z3
C
      DATA MACHEP/Z3410000000000000/
C
      IERR = 0
      NORM = 0.0D0
      K = 1
C     :::::::::: STORE ROOTS ISOLATED BY BALANC
C                AND COMPUTE MATRIX NORM ::::::::::
      DO 50 I = 1, N
C
         DO 40 J = K, N
   40    NORM = NORM + DABS(H(I,J))
C
         K = I
         IF (I .GE. LOW .AND. I .LE. IGH) GO TO 50
         WR(I) = H(I,I)
         WI(I) = 0.0D0
   50 CONTINUE
C
      EN = IGH
      T = 0.0D0
C     :::::::::: SEARCH FOR NEXT EIGENVALUES ::::::::::
   60 IF (EN .LT. LOW) GO TO 340
      ITS = 0
      NA = EN - 1
      ENM2 = NA - 1
C     :::::::::: LOOK FOR SINGLE SMALL SUB-DIAGONAL ELEMENT
C                FOR L=EN STEP -1 UNTIL LOW DO -- ::::::::::
   70 DO 80 LL = LOW, EN
         L = EN + LOW - LL
         IF (L .EQ. LOW) GO TO 100
         S = DABS(H(L-1,L-1)) + DABS(H(L,L))
         IF (S .EQ. 0.0D0) S = NORM
         IF (DABS(H(L,L-1)) .LE. MACHEP * S) GO TO 100
   80 CONTINUE
C     :::::::::: FORM SHIFT ::::::::::
  100 X = H(EN,EN)
      IF (L .EQ. EN) GO TO 270
      Y = H(NA,NA)
      W = H(EN,NA) * H(NA,EN)
      IF (L .EQ. NA) GO TO 280
      IF (ITS .EQ. 30) GO TO 1000
      IF (ITS .NE. 10 .AND. ITS .NE. 20) GO TO 130
C     :::::::::: FORM EXCEPTIONAL SHIFT ::::::::::
      T = T + X
C
      DO 120 I = LOW, EN
  120 H(I,I) = H(I,I) - X
C
      S = DABS(H(EN,NA)) + DABS(H(NA,ENM2))
      X = 0.75D0 * S
      Y = X
      W = -0.4375D0 * S * S
  130 ITS = ITS + 1
```

```
C     :::::::::: LOOK FOR TWO CONSECUTIVE SMALL
C                SUB-DIAGONAL ELEMENTS.
C                FOR M=EN-2 STEP -1 UNTIL L DO -- ::::::::::
      DO 140 MM = L, ENM2
         M = ENM2 + L - MM
         ZZ = H(M,M)
         R = X - ZZ
         S = Y - ZZ
         P = (R * S - W) / H(M+1,M) + H(M,M+1)
         Q = H(M+1,M+1) - ZZ - R - S
         R = H(M+2,M+1)
         S = DABS(P) + DABS(Q) + DABS(R)
         P = P / S
         Q = Q / S
         R = R / S
         IF (M .EQ. L) GO TO 150
         IF (DABS(H(M,M-1)) * (DABS(Q) + DABS(R)) .LE. MACHEP * DABS(P)
     X    * (DABS(H(M-1,M-1)) + DABS(ZZ) + DABS(H(M+1,M+1)))) GO TO 150
  140 CONTINUE
C
  150 MP2 = M + 2
C
      DO 160 I = MP2, EN
         H(I,I-2) = 0.0D0
         IF (I .EQ. MP2) GO TO 160
         H(I,I-3) = 0.0D0
  160 CONTINUE
C     :::::::::: DOUBLE QR STEP INVOLVING ROWS L TO EN AND
C                COLUMNS M TO EN ::::::::::
      DO 260 K = M, NA
         NOTLAS = K .NE. NA
         IF (K .EQ. M) GO TO 170
         P = H(K,K-1)
         Q = H(K+1,K-1)
         R = 0.0D0
         IF (NOTLAS) R = H(K+2,K-1)
         X = DABS(P) + DABS(Q) + DABS(R)
         IF (X .EQ. 0.0D0) GO TO 260
         P = P / X
         Q = Q / X
         R = R / X
  170    S = DSIGN(DSQRT(P*P+Q*Q+R*R),P)
         IF (K .EQ. M) GO TO 180
         H(K,K-1) = -S * X
         GO TO 190
  180    IF (L .NE. M) H(K,K-1) = -H(K,K-1)
  190    P = P + S
         X = P / S
         Y = Q / S
         ZZ = R / S
         Q = Q / P
         R = R / P
C     :::::::::: ROW MODIFICATION ::::::::::
         DO 210 J = K, N
            P = H(K,J) + Q * H(K+1,J)
            IF (.NOT. NOTLAS) GO TO 200
            P = P + R * H(K+2,J)
            H(K+2,J) = H(K+2,J) - P * ZZ
  200       H(K+1,J) = H(K+1,J) - P * Y
            H(K,J) = H(K,J) - P * X
  210    CONTINUE
C
         J = MIN0(EN,K+3)
C     :::::::::: COLUMN MODIFICATION ::::::::::
         DO 230 I = 1, J
            P = X * H(I,K) + Y * H(I,K+1)
            IF (.NOT. NOTLAS) GO TO 220
            P = P + ZZ * H(I,K+2)
            H(I,K+2) = H(I,K+2) - P * R
  220       H(I,K+1) = H(I,K+1) - P * Q
            H(I,K) = H(I,K) - P
  230    CONTINUE
```

```
C     :::::::::: ACCUMULATE TRANSFORMATIONS ::::::::::
         DO 250 I = LOW, IGH
            P = X * Z(I,K) + Y * Z(I,K+1)
            IF (.NOT. NOTLAS) GO TO 240
            P = P + ZZ * Z(I,K+2)
            Z(I,K+2) = Z(I,K+2) - P * R
  240       Z(I,K+1) = Z(I,K+1) - P * Q
            Z(I,K) = Z(I,K) - P
  250    CONTINUE
C
  260 CONTINUE
C
      GO TO 70
C     :::::::::: ONE ROOT FOUND ::::::::::
  270 H(EN,EN) = X + T
      WR(EN) = H(EN,EN)
      WI(EN) = 0.0D0
      EN = NA
      GO TO 60
C     :::::::::: TWO ROOTS FOUND ::::::::::
  280 P = (Y - X) / 2.0D0
      Q = P * P + W
      ZZ = DSQRT(DABS(Q))
      H(EN,EN) = X + T
      X = H(EN,EN)
      H(NA,NA) = Y + T
      IF (Q .LT. 0.0D0) GO TO 320
C     :::::::::: REAL PAIR ::::::::::
      ZZ = P + DSIGN(ZZ,P)
      WR(NA) = X + ZZ
      WR(EN) = WR(NA)
      IF (ZZ .NE. 0.0D0) WR(EN) = X - W / ZZ
      WI(NA) = 0.0D0
      WI(EN) = 0.0D0
      X = H(EN,NA)
      S = DABS(X) + DABS(ZZ)
      P = X / S
      Q = ZZ / S
      R = DSQRT(P*P+Q*Q)
      P = P / R
      Q = Q / R
C     :::::::::: ROW MODIFICATION ::::::::::
      DO 290 J = NA, N
         ZZ = H(NA,J)
         H(NA,J) = Q * ZZ + P * H(EN,J)
         H(EN,J) = Q * H(EN,J) - P * ZZ
  290 CONTINUE
C     :::::::::: COLUMN MODIFICATION ::::::::::
      DO 300 I = 1, EN
         ZZ = H(I,NA)
         H(I,NA) = Q * ZZ + P * H(I,EN)
         H(I,EN) = Q * H(I,EN) - P * ZZ
  300 CONTINUE
C     :::::::::: ACCUMULATE TRANSFORMATIONS ::::::::::
      DO 310 I = LOW, IGH
         ZZ = Z(I,NA)
         Z(I,NA) = Q * ZZ + P * Z(I,EN)
         Z(I,EN) = Q * Z(I,EN) - P * ZZ
  310 CONTINUE
C
      GO TO 330
C     :::::::::: COMPLEX PAIR ::::::::::
  320 WR(NA) = X + P
      WR(EN) = X + P
      WI(NA) = ZZ
      WI(EN) = -ZZ
  330 EN = ENM2
      GO TO 60
C     :::::::::: ALL ROOTS FOUND.  BACKSUBSTITUTE TO FIND
C                VECTORS OF UPPER TRIANGULAR FORM ::::::::::
  340 IF (NORM .EQ. 0.0D0) GO TO 1001
C     :::::::::: FOR EN=N STEP -1 UNTIL 1 DO -- ::::::::::
```

```
      DO 800 NN = 1, N
         EN = N + 1 - NN
         P = WR(EN)
         Q = WI(EN)
         NA = EN - 1
         IF (Q) 710, 600, 800
C     :::::::::: REAL VECTOR ::::::::::
  600    M = EN
         H(EN,EN) = 1.0D0
         IF (NA .EQ. 0) GO TO 800
C     :::::::::: FOR I=EN-1 STEP -1 UNTIL 1 DO -- ::::::::::
         DO 700 II = 1, NA
            I = EN - II
            W = H(I,I) - P
            R = H(I,EN)
            IF (M .GT. NA) GO TO 620
C
            DO 610 J = M, NA
  610       R = R + H(I,J) * H(J,EN)
C
  620       IF (WI(I) .GE. 0.0D0) GO TO 630
            ZZ = W
            S = R
            GO TO 700
  630       M = I
            IF (WI(I) .NE. 0.0D0) GO TO 640
            T = W
            IF (W .EQ. 0.0D0) T = MACHEP * NORM
            H(I,EN) = -R / T
            GO TO 700
C     :::::::::: SOLVE REAL EQUATIONS ::::::::::
  640       X = H(I,I+1)
            Y = H(I+1,I)
            Q = (WR(I) - P) * (WR(I) - P) + WI(I) * WI(I)
            T = (X * S - ZZ * R) / Q
            H(I,EN) = T
            IF (DABS(X) .LE. DABS(ZZ)) GO TO 650
            H(I+1,EN) = (-R - W * T) / X
            GO TO 700
  650       H(I+1,EN) = (-S - Y * T) / ZZ
  700    CONTINUE
C     :::::::::: END REAL VECTOR ::::::::::
         GO TO 800
C     :::::::::: COMPLEX VECTOR ::::::::::
  710    M = NA
C     :::::::::: LAST VECTOR COMPONENT CHOSEN IMAGINARY SO THAT
C                EIGENVECTOR MATRIX IS TRIANGULAR ::::::::::
         IF (DABS(H(EN,NA)) .LE. DABS(H(NA,EN))) GO TO 720
         H(NA,NA) = Q / H(EN,NA)
         H(NA,EN) = -(H(EN,EN) - P) / H(EN,NA)
         GO TO 730
  720    Z3 = DCMPLX(0.0D0,-H(NA,EN)) / DCMPLX(H(NA,NA)-P,Q)
         H(NA,NA) = DREAL(Z3)
         H(NA,EN) = DIMAG(Z3)
  730    H(EN,NA) = 0.0D0
         H(EN,EN) = 1.0D0
         ENM2 = NA - 1
         IF (ENM2 .EQ. 0) GO TO 800
C     :::::::::: FOR I=EN-2 STEP -1 UNTIL 1 DO -- ::::::::::
         DO 790 II = 1, ENM2
            I = NA - II
            W = H(I,I) - P
            RA = 0.0D0
            SA = H(I,EN)
C
            DO 760 J = M, NA
               RA = RA + H(I,J) * H(J,NA)
               SA = SA + H(I,J) * H(J,EN)
  760       CONTINUE
C
            IF (WI(I) .GE. 0.0D0) GO TO 770
            ZZ = W
```

```
            R = RA
            S = SA
            GO TO 790
  770       M = I
            IF (WI(I) .NE. 0.0D0) GO TO 780
            Z3 = DCMPLX(-RA,-SA) / DCMPLX(W,Q)
            H(I,NA) = DREAL(Z3)
            H(I,EN) = DIMAG(Z3)
            GO TO 790
C     :::::::::: SOLVE COMPLEX EQUATIONS ::::::::::
  780       X = H(I,I+1)
            Y = H(I+1,I)
            VR = (WR(I) - P) * (WR(I) - P) + WI(I) * WI(I) - Q * Q
            VI = (WR(I) - P) * 2.0D0 * Q
            IF (VR .EQ. 0.0D0 .AND. VI .EQ. 0.0D0) VR = MACHEP * NORM
     X       * (DABS(W) + DABS(Q) + DABS(X) + DABS(Y) + DABS(ZZ))
            Z3 = DCMPLX(X*R-ZZ*RA+Q*SA,X*S-ZZ*SA-Q*RA) / DCMPLX(VR,VI)
            H(I,NA) = DREAL(Z3)
            H(I,EN) = DIMAG(Z3)
            IF (DABS(X) .LE. DABS(ZZ) + DABS(Q)) GO TO 785
            H(I+1,NA) = (-RA - W * H(I,NA) + Q * H(I,EN)) / X
            H(I+1,EN) = (-SA - W * H(I,EN) - Q * H(I,NA)) / X
            GO TO 790
  785       Z3 = DCMPLX(-R-Y*H(I,NA),-S-Y*H(I,EN)) / DCMPLX(ZZ,Q)
            H(I+1,NA) = DREAL(Z3)
            H(I+1,EN) = DIMAG(Z3)
  790    CONTINUE
C     :::::::::: END COMPLEX VECTOR ::::::::::
  800 CONTINUE
C     :::::::::: END BACK SUBSTITUTION.
C                VECTORS OF ISOLATED ROOTS ::::::::::
      DO 840 I = 1, N
         IF (I .GE. LOW .AND. I .LE. IGH) GO TO 840
C
         DO 820 J = I, N
  820    Z(I,J) = H(I,J)
C
  840 CONTINUE
C     :::::::::: MULTIPLY BY TRANSFORMATION MATRIX TO GIVE
C                VECTORS OF ORIGINAL FULL MATRIX.
C                FOR J=N STEP -1 UNTIL LOW DO -- ::::::::::
      DO 880 JJ = LOW, N
         J = N + LOW - JJ
         M = MIN0(J,IGH)
C
         DO 880 I = LOW, IGH
            ZZ = 0.0D0
C
            DO 860 K = LOW, M
  860       ZZ = ZZ + Z(I,K) * H(K,J)
C
            Z(I,J) = ZZ
  880 CONTINUE
C
      GO TO 1001
C     :::::::::: SET ERROR -- NO CONVERGENCE TO AN
C                EIGENVALUE AFTER 30 ITERATIONS ::::::::::
 1000 IERR = EN
 1001 RETURN
C     :::::::::: LAST CARD OF HQR2 ::::::::::
      END
C
C     ------------------------------------------------------------
C
      SUBROUTINE BALBAK(NM,N,LOW,IGH,SCALE,M,Z)
C
      INTEGER I,J,K,M,N,II,NM,IGH,LOW
      REAL*8 SCALE(N),Z(NM,M)
      REAL*8 S
C
      IF (M .EQ. 0) GO TO 200
      IF (IGH .EQ. LOW) GO TO 120
```

```
C
      DO 110 I = LOW, IGH
         S = SCALE(I)
C     :::::::::: LEFT HAND EIGENVECTORS ARE BACK TRANSFORMED
C                IF THE FOREGOING STATEMENT IS REPLACED BY
C                S=1.0D0/SCALE(I). ::::::::::
         DO 100 J = 1, M
  100    Z(I,J) = Z(I,J) * S
C
  110 CONTINUE
C     ::::::::: FOR I=LOW-1 STEP -1 UNTIL 1,
C               IGH+1 STEP 1 UNTIL N DO -- ::::::::::
  120 DO 140 II = 1, N
         I = II
         IF (I .GE. LOW .AND. I .LE. IGH) GO TO 140
         IF (I .LT. LOW) I = LOW - II
         K = SCALE(I)
         IF (K .EQ. I) GO TO 140
C
         DO 130 J = 1, M
            S = Z(I,J)
            Z(I,J) = Z(K,J)
            Z(K,J) = S
  130    CONTINUE
C
  140 CONTINUE
C
  200 RETURN
C     :::::::::: LAST CARD OF BALBAK ::::::::::
      END
C
C
C     -----------------------------------------------------------------
C
      SUBROUTINE HQR(NM,N,LOW,IGH,H,WR,WI,IERR)
C
      INTEGER I,J,K,L,M,N,EN,LL,MM,NA,NM,IGH,ITS,LOW,MP2,ENM2,IERR
      REAL*8 H(NM,N),WR(N),WI(N)
      REAL*8 P,Q,R,S,T,W,X,Y,ZZ,NORM,MACHEP
      REAL*8 DSQRT,DABS,DSIGN
      INTEGER MIN0
      LOGICAL NOTLAS
C
      DATA MACHEP/Z3410000000000000/
C
      IERR = 0
      NORM = 0.0D0
      K = 1
C     :::::::::: STORE ROOTS ISOLATED BY BALANC
C                AND COMPUTE MATRIX NORM ::::::::::
      DO 50 I = 1, N
C
         DO 40 J = K, N
   40    NORM = NORM + DABS(H(I,J))
C
         K = I
         IF (I .GE. LOW .AND. I .LE. IGH) GO TO 50
         WR(I) = H(I,I)
         WI(I) = 0.0D0
   50 CONTINUE
C
      EN = IGH
      T = 0.0D0
C     :::::::::: SEARCH FOR NEXT EIGENVALUES ::::::::::
   60 IF (EN .LT. LOW) GO TO 1001
      ITS = 0
      NA = EN - 1
      ENM2 = NA - 1
C     :::::::::: LOOK FOR SINGLE SMALL SUB-DIAGONAL ELEMENT
C                FOR L=EN STEP -1 UNTIL LOW DO -- ::::::::::
   70 DO 80 LL = LOW, EN
         L = EN + LOW - LL
         IF (L .EQ. LOW) GO TO 100
```

```
         S = DABS(H(L-1,L-1)) + DABS(H(L,L))
         IF (S .EQ. 0.0D0) S = NORM
         IF (DABS(H(L,L-1)) .LE. MACHEP * S) GO TO 100
   80 CONTINUE
C     :::::::::: FORM SHIFT ::::::::::
  100 X = H(EN,EN)
      IF (L .EQ. EN) GO TO 270
      Y = H(NA,NA)
      W = H(EN,NA) * H(NA,EN)
      IF (L .EQ. NA) GO TO 280
      IF (ITS .EQ. 30) GO TO 1000
      IF (ITS .NE. 10 .AND. ITS .NE. 20) GO TO 130
C     :::::::::: FORM EXCEPTIONAL SHIFT ::::::::::
      T = T + X
C
      DO 120 I = LOW, EN
  120 H(I,I) = H(I,I) - X
C
      S = DABS(H(EN,NA)) + DABS(H(NA,ENM2))
      X = 0.75D0 * S
      Y = X
      W = -0.4375D0 * S * S
  130 ITS = ITS + 1
C     :::::::::: LOOK FOR TWO CONSECUTIVE SMALL
C                SUB-DIAGONAL ELEMENTS.
C                FOR M=EN-2 STEP -1 UNTIL L DO -- ::::::::::
      DO 140 MM = L, ENM2
         M = ENM2 + L - MM
         ZZ = H(M,M)
         R = X - ZZ
         S = Y - ZZ
         P = (R * S - W) / H(M+1,M) + H(M,M+1)
         Q = H(M+1,M+1) - ZZ - R - S
         R = H(M+2,M+1)
         S = DABS(P) + DABS(Q) + DABS(R)
         P = P / S
         Q = Q / S
         R = R / S
         IF (M .EQ. L) GO TO 150
         IF (DABS(H(M,M-1)) * (DABS(Q) + DABS(R)) .LE. MACHEP * DABS(P)
     X    * (DABS(H(M-1,M-1)) + DABS(ZZ) + DABS(H(M+1,M+1)))) GO TO 150
  140 CONTINUE
C
  150 MP2 = M + 2
C
      DO 160 I = MP2, EN
         H(I,I-2) = 0.0D0
         IF (I .EQ. MP2) GO TO 160
         H(I,I-3) = 0.0D0
  160 CONTINUE
C     :::::::::: DOUBLE QR STEP INVOLVING ROWS L TO EN AND
C                COLUMNS M TO EN ::::::::::
      DO 260 K = M, NA
         NOTLAS = K .NE. NA
         IF (K .EQ. M) GO TO 170
         P = H(K,K-1)
         Q = H(K+1,K-1)
         R = 0.0D0
         IF (NOTLAS) R = H(K+2,K-1)
         X = DABS(P) + DABS(Q) + DABS(R)
         IF (X .EQ. 0.0D0) GO TO 260
         P = P / X
         Q = Q / X
         R = R / X
  170    S = DSIGN(DSQRT(P*P+Q*Q+R*R),P)
         IF (K .EQ. M) GO TO 180
         H(K,K-1) = -S * X
         GO TO 190
  180    IF (L .NE. M) H(K,K-1) = -H(K,K-1)
  190    P = P + S
         X = P / S
         Y = Q / S
```

```
            ZZ = R / S
            Q = Q / P
            R = R / P
C      :::::::::: ROW MODIFICATION ::::::::::
            DO 210 J = K, EN
               P = H(K,J) + Q * H(K+1,J)
               IF (.NOT. NOTLAS) GO TO 200
               P = P + R * H(K+2,J)
               H(K+2,J) = H(K+2,J) - P * ZZ
  200          H(K+1,J) = H(K+1,J) - P * Y
               H(K,J) = H(K,J) - P * X
  210       CONTINUE
C
            J = MIN0(EN,K+3)
C      :::::::::: COLUMN MODIFICATION ::::::::::
            DO 230 I = L, J
               P = X * H(I,K) + Y * H(I,K+1)
               IF (.NOT. NOTLAS) GO TO 220
               P = P + ZZ * H(I,K+2)
               H(I,K+2) = H(I,K+2) - P * R
  220          H(I,K+1) = H(I,K+1) - P * Q
               H(I,K) = H(I,K) - P
  230       CONTINUE
C
  260 CONTINUE
C
      GO TO 70
C      :::::::::: ONE ROOT FOUND ::::::::::
  270 WR(EN) = X + T
      WI(EN) = 0.0D0
      EN = NA
      GO TO 60
C      :::::::::: TWO ROOTS FOUND ::::::::::
  280 P = (Y - X) / 2.0D0
      Q = P * P + W
      ZZ = DSQRT(DABS(Q))
      X = X + T
      IF (Q .LT. 0.0D0) GO TO 320
C      :::::::::: REAL PAIR ::::::::::
      ZZ = P + DSIGN(ZZ,P)
      WR(NA) = X + ZZ
      WR(EN) = WR(NA)
      IF (ZZ .NE. 0.0D0) WR(EN) = X - W / ZZ
      WI(NA) = 0.0D0
      WI(EN) = 0.0D0
      GO TO 330
C      :::::::::: COMPLEX PAIR ::::::::::
  320 WR(NA) = X + P
      WR(EN) = X + P
      WI(NA) = ZZ
      WI(EN) = -ZZ
  330 EN = ENM2
      GO TO 60
C      :::::::::: SET ERROR -- NO CONVERGENCE TO AN
C                 EIGENVALUE AFTER 30 ITERATIONS ::::::::::
 1000 IERR = EN
 1001 RETURN
C      :::::::::: LAST CARD OF HQR ::::::::::
      END
C ---------------------------------------------------------------------------
      SUBROUTINE PSDCAL(M2,NS,FA,X,NC,GW,SV ,C,NO,HY,HU,H,
     1 FBGE,NG,GAM,ACL,P,WR,WI,DT,D2,JCP,RES,Q,E,BB,CC,IYU,
     2 IPSD,INORM)
C
C         PSDCAL COMPUTES THE PSD OF OUTPUTS OR CONTROLS OF
C          A CONTROLLED SYSTEM
C
C           IYU= 1      OUTPUT PSD
C              = 2      CONTROL PSD
C
C           IPSD=1      PSD
C               =2      PSD AND TF RESIDUES
```

```
C
C
C           INORM=      1,2,... NG   NORMALIZED BY ITH PROCESS NOISE
C                       NG+1,... NG+NO NORMALIZED BY ITH MEAS NOISE
C
      DOUBLE PRECISION FA,X,GW,GV,C,HY,H,PBGE,GAM,ACL,F,WR,WI,D1,D2,RES
     1 BB,CC,Q,R,PSD,W,DNORM,DN1,EMAX,ELOG,EMOD,DW,ST,OM,RE,AI,HU,DW1
      COMPLEX*16 ZD,ZN,ZZ
      DIMENSION FA(N2,N2),X(N2,N2),GW(N2,NG),C(NC,NS),HY(NO,N2),
     1 H(NO,NS),PBGE(NS,NO),GAM(NS,NG),ACL(NS,NS),F(NS,NS),WR(N2),
     2 WI(N2),D1(N2),D2(N2),RES(N2),Q(NG,NG),R(NO,NO),PSD(30),
     3 W(30),BB(N2),CC(N2),GV(N2,NO),HU(NC,N2),DW1(4)
      INTEGER JCP(N2)
      DATA DW1/1.D0,2.D0,5.D0,10.D0/
C
      IF(IYU .EQ. 0) IYU=1
      IF(INORM .EQ. 0) INORM = 1
      IPT = 0
      IF(IPSD .GT. 1) IPT = 1
C
      IX = INORM - NG
      IF(IX .GT. 0) WRITE(6,8000) IX
 8000 FORMAT(/' SUBSEQUENT PSD IS NORMALIZED BY MEAS NO.',I3)
      IF(IX .LE. 0) WRITE(6,8010) INORM
 8010 FORMAT(/' SUBSEQUENT PSD IS NORMALIZED BY PROCESS NOISE NO. ',I3)
      NSQ = N2*N2
C
C     :::::::::: COMPUTE EIGENSYSTEM OF CONTROLLED SYSTEM
C
C     :::::::::: FORM FA
      DO 10 I=1,NS
      DO 10 J=1,NS
      FA(I,J) = ACL(I,J)
   10 FA(NS+I,J) = 0.D0
      DO 20 I=1,NS
      DO 20 J=1,NS
      ST = 0.D0
      DO 15 K=1,NO
   15 ST = ST + PBGE(I,K)*H(K,J)
      FA(I,NS+J) = -ST
   20 FA(NS+I,NS+J) = F(I,J) - ST
      CALL RAPRNT(N2,N2,N2,9,FA,4,'(9(1X,1PD13.6))')
C[illegible] DEBUG ABOVE
C     [illegible]
      CALL BALANC(N2,N2,FA,LOW,IHIGH,D1)
      CALL ORTHES(N2,N2,LOW,IHIGH,FA,D2)
      CALL ORTRAN(N2,N2,LOW,IHIGH,FA,D2,X)
      CALL HQR2(N2,N2,LOW,IHIGH,FA,WR,WI,X,IERR)
      IF(IERR .NE. 0) GO TO 1000
      CALL BALBAK(N2,N2,LOW,IHIGH,D1,N2,X)
C     [illegible]
      CALL RAPRNT(N2,N2,N2,9,X,4,'(9(1X,1PD13.6))')
C[illegible] DEBUG ABOVE
  100 CONTINUE
C     :::::::::: DETERMINE MODAL MATRICES
      IF(IYU .EQ. 1) GO TO 130
C     :::::::::: HSUBU
      DO 110 I=1,NC
      DO 110 J=1,N2
      ST = 0.D0
      DO 105 K=1,NS
  105 ST = ST - C(I,K)*X(K,J)
  110 HU(I,J) = ST
      GO TO 150
C     :::::::::: HSUBY
  130 DO 140 I=1,NO
      DO 140 J=1,N2
      ST = 0.D0
      DO 135 K=1,NS
  135 ST = ST + H(I,K)*X(K,J) - H(I,K)*X(NS+K,J)
  140 HY(I,J) = ST
      CALL RAPRNT(NO,NO,N2,9,HY,4,'(9(1X,1PD13.6))')
```

```
C************** DEBUG ABOVE
  150 CALL MINV(NSQ,X,N2,ST,D1,D2)
      CALL RAPRNT(N2,N2,N2,9,X,4,'(9(1X,1PD13.6))')
C************** DEBUG ABOVE
C     :::::::::: GSUBW
      DO 160 I=1,N2
      DO 160 J=1,NG
      ST = 0.0D0
      DO 155 K=1,NS
  155 ST = ST - X(I,NS+K)*GAM(K,J)
  160 GW(I,J) = ST
      CALL RAPRNT(N2,N2,NG,9,GW,4,'(9(1X,1PD13.6))')
C************** DEBUG ABOVE
C     :::::::::: USE SELECTED NORMALIZATION
  200 IF(INORM .LE. NG) DNORM = 1.D0/Q(INORM,INORM)
      IF(INORM .GT. NG) DNORM = 1.D0/R(INORM-NG,INORM-NG)
C     :::::::::: DETERMINE BANDWIDTH OF CONTROLLED SYS
      EMAX = 0.D0
      DO 210 I=1,N2
      EMOD = DABS(WR(I)**2 +WI(I)**2)
      IF(EMOD .GT. EMAX) EMAX = EMOD
  210 CONTINUE
      EMOD = DSQRT(EMAX)
      EMOD=2*EMOD
C     :::::::::: ROUND UP TO NEAREST 2,4,5,8,10
      ELOG = DLOG10(EMOD)
      IF(ELOG .LT. 0.D0) IPOW = -IDINT(DABS(ELOG) + 1)
      IF(ELOG .GE. 0.D0) IPOW =  IDINT(ELOG)
      EMAX = EMOD*10**(-IPOW)
      IF(EMAX .GT. 2.D0) EMOD = 2.D0
      IF(EMAX .GT. 4.D0) EMOD = 4.D0
      IF(EMAX .GT. 5.D0) EMOD = 5.D0
      IF(EMAX .GT. 8.D0) EMOD = 8.D0
      IF(EMAX .GE. 10.D0) EMOD = 10.D0
      EMAX = EMOD*10**IPOW
      DW = EMAX/20.D0
C     :::::::::: ADD 10 POINTS 3 DECADES UP
      IF(EMOD .LT. 5.0) GO TO 212
      EMAX = 1.0D1
      IK = 3
      GO TO 216
  212 EMAX = 5.D0
      IK = 2
  216 CONTINUE
C     :::::::::: STORE 30 FREQUENCIES
      DO 220 I=1,20
  220 W(I) = DW*(I-1)
      DO 218 I=1,3
        IP = 20 + 3*(I-1)
        DO 218 J=1,3
          IX = MOD (IK+J-1,3) + 1
          JJ = 0
          IF(IK .EQ. 2 .AND. J .GE. 2) JJ=1
          W(IP+J) = DW1(IX)*10**(IPOW+I-1+JJ+IK-2)
  218 CONTINUE
      IX = MOD(IK,3) + 1
      W(30) = DW1(IX)*10**(IPOW+3 +IK-2)
C     :::::::::: LARGE LOOP THRU OUTPUTS
      IF(IYU .EQ. 1) NL= NO
      IF(IYU .EQ. 2) NL = NC
      DO 400 L=1,NL
        DO 250 I=1,30
  250   FSD(I) = 0.D0
C     :::::::::: LOOP THRU PROCESS NOISE
        DO 300 I=1,NG
          DW1 = DNORM*Q(I,I)
      IF(IYU .EQ. 1 .AND. IPT .EQ. 1) WRITE(6,8020) I,L
 8020 FORMAT(/' TRANSFER FUNCTION FROM PROCESS NOISE ',I2,' TO'
     1,' MEASUREMENT ',I2)
      IF(IYU .EQ. 2 .AND. IPT .EQ. 1) WRITE(6,8030) I,L
 8030 FORMAT(/' TRANSFER FUNCTION FROM PROCESS NOISE ',I2,' TO',
     1' CONTROL ',I2)
```

```
      IF(IYU.EQ.1)CALL RESID(I,L,N2,JCP,NG,GW,NL,HY,WR,WI,
     1 RES,BB,CC,IPT)
      IF(IYU.EQ.2)CALL RESID(I,L,N2,JCP,NG,GW,NL,HU,WR,WI,
     1 RES,BB,CC,IPT)
      DO 280 K=1,20
        ZZ = DCMPLX(0.D0,0.D0)
         OM = W(K)
         DO 260 II=1,N2
            IF(WI(II)) 260,254,256
  254       ZD = DCMPLX(-WR(II),OM-WI(II))
            ZZ = RES(II)/ZD + ZZ
            GO TO 260
  256       RE = WR(II)
            AI = WI(II)
            ZD = DCMPLX(RE**2 + AI**2 - OM**2,-2.D0*RE*OM)
            ZN = DCMPLX(RES(II+1)*AI-RES(II)*RE,RES(II)*OM)
            ZZ = ZZ + ZN/ZD
  260       CONTINUE
  280    PSD(K) = PSD(K) + DN1*(ZZ*DCONJG(ZZ))
  300 CONTINUE
C     :::::::::: GSUBV
      DO 320 I=1,N2
      DO 320 J=1,NO
      ST = 0.D0
      DO 315 K=1,NS
  315 ST = ST + X(I,K)*FBGE(K,J) + X(I,NS+K)*FBGE(K,J)
  320 GV(I,J) = ST
      CALL RAPRNT(N2,N2,NO,9,GV,3,'(9(1X,1PD13.6))')
C********************* DEBUG ABOVE
C     :::::::::: LOOP THRU MEAS NOISE
      DO 390 I=1,NO
         DN1 = DNORM*R(I,I)
      IF(IYU .EQ. 1 .AND. IPT .EQ. 1) WRITE(6,8040) I,L
 8040 FORMAT(/' TRANSFER FUNCTION FROM MEASUREMENT ',I2,
     1 ' TO MEASUREMENT ',I2)
      IF(IYU .EQ. 2 .AND. IPT .EQ. 1) WRITE(6,8050) I,L
 8050 FORMAT(/' TRANSFER FUNCTION FROM MEASUREMENT ',I2,
     1 ' TO CONTROL ',I2)
        IF(IYU.EQ.1)CALL RESID(I,L,N2,JCP,NO,GV,NL,HY,WR,WI,RES,
     1   BB,CC,IPT)
        IF(IYU.EQ.2)CALL RESID(I,L,N2,JCP,NO,GV,NL,HU,WR,WI,RES,
     1   BB,CC,IPT)
        DO 380 K=1,30
           ZZ = DCMPLX(0.D0,0.D0)
           OM = W(K)
           DO 360 II=1,N2
              IF(WI(II)) 360,354,356
  354         ZD = DCMPLX(-WR(II),OM-WI(II))
              ZZ = ZZ + RES(II)/ZD
              GO TO 360
  356         RE=WR(II)
              AI = WI(II)
              ZD = DCMPLX(RE**2 + AI**2 -OM**2,-2.D0*RE*OM)
              ZN = DCMPLX(RES(II+1)*AI-RES(II)*RE,RES(II)*OM)
              ZZ = ZZ + ZN/ZD
  360         CONTINUE
         IF(IYU .EQ. 2 .OR. I .NE. L) GO TO 378
          PSD(K) = PSD(K) + DN1
  378     PSD(K) = PSD(K) + DN1*(ZZ*DCONJG(ZZ))
  380     CONTINUE
  390     CONTINUE
      IF(IYU .EQ. 1) WRITE(6,9000) L
      IF(IYU .EQ. 2) WRITE(6,9010) L
 9000 FORMAT(/' PSD OF OUTPUT',I3,' FORCED BY ALL NOISE-(RAD FREQ,'
     1    'NORMALIZED PSD)'/)
 9010 FORMAT(/' PSD OF CONTROL',I3,' FORCED BY ALL NOISE-(RAD FREQ,'
     1    'NORMALIZED PSD)'/)
      WRITE(6,9020) (W(I),PSD(I),I=1,30)
 9020 FORMAT(4(1X,'(',E11.4,',',E11.4,')'))
  400 CONTINUE
      RETURN
 1000 CONTINUE
```

```
      CALL EREXIT(N2,FA,IERR)
      RETURN
      END
C ----------------------------------------------------------------------
      SUBROUTINE EREXIT(N,A,IERR)
C
C       EREXIT RETURNS THE NUMBER OF THE EIGENVALUE WHERE HQR2
C        FAILS, THEN STOPS THE PROGRAM.
C
      INTEGER IERR
      DOUBLE PRECISION A
      DIMENSION A(N,N)
      WRITE(6,9000) IERR
 9000 FORMAT(' FAILURE IN HQR2 ON EIGENVALUE NO. ',I3)
      CALL RAPENT(N,N,N,9,A,4,'(9(1X,1PD13.6))')
      STOP
      END
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                             C
C              SENSITIVITY COVARIANCE PROGRAM                 C
C                                                             C
C      THIS PROGRAM IS USED TO SOLVE THE SET OF SENSITIVITY   C
C      EQUATIONS WHEN THERE IS AN INCORRECT IMPLEMENTATION    C
C      OF DYNAMICS IN THE DESIGN OF THE KALMAN FILTER.THE     C
C      EQUATIONS BECOME                                       C
C                                                             C
C      PDOT=(F*-K*H)P+P(F*-K*H)T+DPV+VTDPT+GQGT+K*RK*T        C
C      VDOT=FV+V(F*-K*H)T+UDFT-GQGT                           C
C      UDOT=FU+UFT+GQGT                                       C
C                                                             C
C      THE PRINCIPAL PROGRAM INPUTS ARE THE FOLLOWING CO-     C
C      LLECTION OF SYSTEM AND FILTER MATRICES                 C
C                                                             C
C      PO    THE INITIAL COVARIANCE MATRIX(NXN)               C
C      F     THE TRUTH MODEL DYNAMICS MATRIX(NXN)             C
C      F*    THE FILTER MODEL DYNAMICS MATRIX(NXN)            C
C      H     THE TRUTH MODEL MEASUREMENT MATRIX(LXN),WHERE    C
C            L IS THE MEASUREMENT VECTOR DIMENSION            C
C      GQGT  THE INPUT NOISE COVARIANCE MATRIX(NX)            C
C      R     THE MEASUREMENT NOISE COVARIANCE MATRIX(LXL)     C
C      K*    FILTER GAIN(NXL)                                 C
C                                                             C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
C      THIS PROGRAM HAS BEEN DEVELOPED USING THE IMSL LIBRARY
C      AVAILABLE IN THE COMPUTER CENTER OF THE NAVAL
C      POSTGRADUATE SCHOOL
C
C
       IMPLICIT REAL*8 (A-H,O-Z)
       COMMON F(7,7),FS(7,7),GQGT(7),AK(7,2),R(2,2),
      *AKRKT(7,7),DF(7,7),FT(7,7),FSAKHT(7,7),DFT(7,7),
      *FSAKH(7,7),H(2,7)
       COMMON/KTR/N,NS,NPD
       DIMENSION PFULL(7,7),PSQR(7)
       DIMENSION DDT(7)
       DIMENSION U(28),V(7,7),P(28),UD(28),VD(7,7),
      *VAR(105),DRV(105),C(24),WK(105,9),PD(28)
       DIMENSION TMP1(7,7),TMP2(7,7),TMP3(7,7)
       EQUIVALENCE (U(1),VAR(1)),(V(1,1),VAR(29)),(P(1),
      *VAR(78)),(UD(1),DRV(1)),(VD(1,1),DRV(29)),(PD(1),
      *DRV(78))
C
C      N=ORDER OF THE SYSTEM MODEL
C
C      NP=NUMBER OF POINTS
C
C      NPD=CONTROL OF INITIAL DIAGNOSTIC OUTPUT
C
C      DT=TIME INTERVAL
C
       EXTERNAL FUN
       CALL UGETIO(3,5,6)
C
C
C      THE FOLLOWING SECTION READS THE SPECIFIED INPUT
C      MATRICES,F,F*,GQGT,K*R AND R
C
C
       READ(5,98)N,NP,NPD,DT
   98  FORMAT(3I5,F10.5)
       WRITE(6,97)N,NP,NPD,DT
   97  FORMAT('1',3I5,1X,G16.10)
       NS=N*(N+1)/2
       N1=NS+N**2+1
       NV=2*NS+N**2
   99  FORMAT(8F10.5)
C
```

```
      DO 1 I=1,N
    1 READ(5,99) (F(I,J),J=1,N)
      CALL USWFM('F',1,F,7,N,N,1)
C
      DO 2 I=1,N
    2 READ(5,99) (FS(I,J),J=1,N)
      CALL USWFM('FS',2,FS,7,N,N,1)
      READ(5,99) (GQGT(I),I=1,N)
      CALL USWFV('GQGT',4,GQGT,N,1,1)
C
      DO 3 I=1,N
    3 READ(5,99) (AK(I,J),J=1,2)
      CALL USWFM('K',1,AK,7,N,2,1)
C
      DO 4 I=1,2
    4 READ(5,99) (H(I,J),J=1,N)
      CALL USWFM('H',1,H,2,2,N,1)
C
      DO 5 I=1,2
    5 READ(5,99) (R(I,J),J=1,2)
      CALL USWFM('R',1,R,2,2,2,1)
C
      DO 6 I=1,105
    6 VAR(I)=0.
      DO 7 I=1,N
      DO 7 J=1,N
C
    7 DF(I,J)=FS(I,J)-F(I,J)
      CALL USWFM('DEL F',5,DF,7,N,N,1)
      CALL VMULFF(AK,R,N,2,2,7,2,TMP1,7,IER)
      CALL VMULFP(TMP1,AK,N,2,N,7,7,AKRKT,7,IER)
      CALL USWFM('KRKT',4,AKRKT,7,N,N,1)
C
C     CALCULATE THE DIFFERENCE BETWEEN THE DYNAMICS
C     IMPLEMENTED IN THE FILTER AND THE PLANT, DF=FS-F
C
      DO 20 I=1,N
      DO 20 J=1,N
      DFT(I,J)=DF(I,J)
   20 FT(I,J)=F(I,J)
      CALL VTRANX(DFT,N,N,7)
      CALL USWFM('DEL FT',6,DFT,7,N,N,1)
      CALL VTRANX(FT,N,N,7)
      CALL USWFM('FT',2,FT,7,N,N,1)
      CALL VMULFF(AK,H,N,2,N,7,2,TMP1,7,IER)
C
      DO 21 I=1,7
      DO 21 J=1,7
      FSMKH(I,J)=FS(I,J)-TMP1(I,J)
   21 FSMKHT(I,J)=FSMKH(I,J)
      CALL USWFM('FS-KH',5,FSMKH,7,N,N,1)
      CALL VTRANX(FSMKHT,N,N,7)
      CALL USWFM('(FS-KH)T',8,FSMKHT,7,N,N,1)
      T=0.
      TOL=1.D-5
      IND=1
      L=0
      IF(N.EQ.7) GO TO 11
C
      DO 31 I=1,NS
      L=L+1
   31 VAR(L)=U(I)
C
      DO 32 I=1,N
      DO 32 J=1,N
      L=L+1
   32 VAR(L)=V(I,J)
C
      DO 33 I=1,N
      L=L+1
   33 VAR(L)=P(I)
   11 DO 10 K=1,NP
```

```
C     TEND=FINAL TIME
C
      TEND=K*DT
C
C     DVERK SUBROUTINE FINDS THE SOLUTION OF THE SYSTEM OF
C     DIFFERENTIAL EQUATIONS
C
      CALL DVERK(NV,FUN,T,VAR,TEND,TOL,IND,C,105,WK,IER)
      IF(IND.LE.0.OR.IER.NE.0) STOP
      CALL VCVTSF(VAR(N1),N,PFULL,7)
C
C     CALCULATE AND PRINT THE RMS ESTIMATE ERRORS
C
      DO 30 I=1,N
      REAP=PFULL(I,I)
      PFULL(I,I)=DABS(REAP)
   30 PSQR(I)=DSQRT(PFULL(I,I))
      WRITE(6,90)T,(PSQR(I),I=1,N)
   90 FORMAT('0T=',F10.5,' PSR=',7G15.7)
C
C     IF DESIRED PRINT THE  COVARIANCE MATRICES,P,U AND V
C     CALL USWSM('U',1,U,N,2)
C     CALL USWFM('V',1,VAR(NS+1),7,N,N,2)
C     CALL USWSM('P',1,VAR(N1),N,2)
   10 CONTINUE
      STOP
      END
      SUBROUTINE VTRANX(A,N,NC,IA)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(IA,IA),B(7,7)
C
      DO 1 I=1,N
      DO 1 J=1,N
    1 B(I,J)=A(J,I)
C
      DO 2 I=1,N
      DO 2 J=1,N
    2 A(I,J)=B(I,J)
      RETURN
      END
      SUBROUTINE FUN(NV,T,VAR,DRV)
C
C     FCN SUBROUTINE IS USED FOR EVALUATING FUNCTIONS(INPUT)
C
C
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON F(7,7),FS(7,7),GQGT(7,7),AK(7,2),R(2,2),
     *AKRKT(7,7),DF(7,7),FT(7,7),PSRKHT(7,7),DFT(7,7),
     *PSRKH(7,7),H(2,7)
      COMMON/KTR/N,NS,NPD
      DIMENSION U(28),V(7,7),P(28),UD(28),VD(7,7),
     *VAR(105),DRV(105),C(24),WK(105,9),PD(28)
      DIMENSION TMP1(7,7),TMP2(7,7),TMP3(7,7)
C
      L=0
      DO 1 I=1,NS
      L=L+1
    1 U(I)=VAR(L)
C
      DO 2 I=1,N
      DO 2 J=1,N
      L=L+1
    2 V(I,J)=VAR(L)
C
      DO 3 I=1,NS
      L=L+1
    3 P(I)=VAR(L)
      IF(T.EQ.0) KT=NPD
      KT=KT+1
      IF (KT.GT.5) GO TO 15
      WRITE(6,99) T
   99 FORMAT('0T=',D25.15)
```

```
      CALL  USWSM('U',1,U,N,2)
      CALL  USWFM('V',1,V,7,N,N,2)
      CALL  USWSM('P',1,V,N,2)
   15 CALL  VMULFS(F,U,N,N,7,TMP1,7)
      IF(KT.LT.5)  CALL  USWFM('FU',2,TMP1,7,N,N,2)
      CALL  VMULSF(U,N,FT,N,7,TMP2,7)
      IF(KT.LT.5)  CALL  USWFM('UFT',3,TMP2,7,N,N,2)
C
      DO 5  I=1,N
      DO 4  J=1,N
    4 TMP1(I,J)=TMP1(I,J)+TMP2(I,J)
    5 TMP1(I,I)=TMP1(I,I)+GOGT(I)
      CALL  VCVTFS(TMP1,N,7,UD)
      IF(KT.LT.5)  CALL  USWSM('UDOT',4,UD,N,2)
      CALL  VMULFF(F,V,N,N,N,7,7,TMP1,7,IER)
      IF(KT.LT.5)  CALL  USWFM('FV',2,TMP1,7,N,N,2)
      CALL  VMULFF(V,FSMKHT,N,N,N,7,7,TMP2,7,IER)
      IF(KT.LT.5)  CALL  USWFM('V(FS-KH)T',10,TMP2,7,N,N,2)
      CALL  VMULSF(U,N,DFT,N,7,TMP3,7)
      IF(KT.LT.5)  CALL  USWFM('U*DFT',5,TMP3,7,N,N,2)
C
      DO 7  I=1,N
      DO 6  J=1,N
    6 VD(I,J)=TMP1(I,J)+TMP2(I,J)+TMP3(I,J)
    7 VD(I,I)=VD(I,I)-GOGT(I)
      IF(KT.LT.5)  CALL  USWFM('VDOT',4,VD,7,N,N,2)
       CALL  VMULFS(FSMKH,P,N,N,7,TMP1,7)
      IF(KT.LT.5)  CALL  USWFM('(FS-KH)P',8,TMP1,7,N,N,2)
      CALL  VMULSF(P,N,FSMKHT,N,7,TMP2,7)
      IF(KT.LT.5)  CALL  USWFM('P(FS-KH)T',9,TMP2,7,N,N,2)
      CALL  VMULFF(DF,V,N,N,N,7,7,TMP3,7,IER)
      IF(KT.LT.5)  CALL  USWFM('DFV',4,TMP3,7,N,N,2)
C
      DO 8  I=1,N
      DO 8  J=1,N
    8 TMP1(I,J)=TMP1(I,J)+TMP2(I,J)+TMP3(I,J)
      CALL  VMULFM(V,DF,N,N,N,7,7,TMP3,7,IER)
      IF(KT.LT.5)  CALL  USWFM('VT*DF',5,TMP3,7,N,N,2)
C
      DO 10  I=1,N
      DO 9  J=1,N
    9 TMP3(I,J)=TMP1(I,J)+TMP3(I,J)+AKRKT(I,J)
   10 TMP3(I,I)=TMP3(I,I)+GOGT(I)
      IF(KT.LT.5)  CALL  USWFM('PDOT',4,TMP3,7,N,N,2)
      CALL  VCVTFS(TMP3,N,7,PD)
      IF(KT.LT.5)  CALL  USWFV('PDOT(SYM)',9,PD,N,1,2)
      L=0
C
      DO 11  I=1,NS
      L=L+1
   11 DRV(L)=UD(I)
C
      DO 12  I=1,N
      DO 12  J=1,N
      L=L+1
   12 DRV(L)=VD(I,J)
C
      DO 13  I=1,NS
      L=L+1
   13 DRV(L)=PD(I)
      IF(KT.LT.5)  CALL  USWFV('DRV',3,DRV,NV,1,2)
      RETURN
      END
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                          C
C              SENSITIVITY COVARIANCE PROGRAM              C
C                                                          C
C      THIS PROGRAM IS USED TO SOLVE THE ERROR SENSITIVITY C
C      EQUATIONS WHEN THERE IS AN INCORRECT IMPLEMENTATION C
C      OF DYNAMICS IN THE DESIGN OF THE KALMAN FILTER.THE  C
C      EQUATIONS BECOME                                    C
C                                                          C
C      PDOT=(F*-K*H)P+P(F*-K*H)T+DFV+VTDFT+GQGT+K*RK*T     C
C      VDOT=FV+V(F*-K*H)T+UDFT-GQGT                        C
C      UDOT=FU+UFT+GQGT                                    C
C                                                          C
C      THE PRINCIPAL PROGRAM INPUTS ARE THE FOLLOWING CO-  C
C      LLECTION OF SYSTEM AND FILTER MATRICES              C
C                                                          C
C      PO    THE INITIAL COVARIANCE MATRIX(NXN)            C
C      F     THE TRUTH MODEL DYNAMICS MATRIX(NXN)          C
C      F*    THE FILTER MODEL DYNAMICS MATRIX(NXN)         C
C      H     THE TRUTH MODEL MEASUREMENT MATRIX(LXN),WHERE C
C            L IS THE MEASUREMENT VECTOR DIMENSION         C
C      GQGT  THE INPUT NOISE COVARIANCE MATRIX(NX)         C
C      R     THE MEASUREMENT NOISE COVARIANCE MATRIX(LXL)  C
C      K*    FILTER GAIN(NXL)                              C
C                                                          C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
C      THIS PROGRAM HAS BEEN DEVELOPED USING THE IMSL LIBRARY
C      AVAILABLE IN THE COMPUTER CENTER OF THE NAVAL
C      POSTGRADUATE SCHOOL
C
C
       IMPLICIT REAL*8 (A-H,O-Z)
       COMMON P(8,8),PS(8,8),GQGT(8),AK(8,3),R(3,3),
      *AKRKT(8,8),DF(8,8),FT(8,8),FSMKHT(8,8),DFT(8,8),
      *FSMKH(8,8),H(3,8)
       COMMON/KTR/N,NS,NPD
       DIMENSION FFULL(8,8),PSQR(8)
       DIMENSION DDT(8)
       DIMENSION U(36),V(8,8),P(36),UD(36),VD(8,8),
      *VAR(136),DRV(136),C(24),WK(136,9),PD(36)
       DIMENSION TMP1(8,8),TMP2(8,8),TMP3(8,8)
       EQUIVALENCE (U(1),VAR(1)),(V(1,1),VAR(37)),(P(1),
      *VAR(101)),(UD(1),DRV(1)),(VD(1,1),DRV(37)),(PD(1),
      *DRV(101))
C
C      N=ORDER OF THE SYSTEM MODEL
C
C      NP=NUMBER OF POINTS
C
C      NPD=CONTROL OF INITIAL DIAGNOSTIC OUTPUT
C
C      DT=TIME INTERVAL
C
       EXTERNAL FUN
       CALL UGETIO(3,5,6)
C
C
C      THE FOLLOWING SECTION READS THE SPECIFIED INPUT
C      MATRICES,F,F*,GQGT,K*H AND R
C
C
       READ(5,98) N,NP,NPD,DT
    98 FORMAT(3I5,F10.5)
       WRITE(6,97)N,NP,NPD,DT
    97 FORMAT('1',3I5,1X,G16.10)
       NS=N*(N+1)/2
       N1=NS+N**2+1
       NV=2*NS+N**2
    99 FORMAT(8F10.5)
C
```

```
      DO 1 I=1,N
    1 READ(5,99) (F(I,J),J=1,N)
      CALL USWFM('F',1,F,8,N,N,1)
C
      DO 2 I=1,N
    2 READ(5,99) (FS(I,J),J=1,N)
      CALL USWFM('FS',2,FS,8,N,N, 1)
      READ(5,99) (GQGT(I),I=1,N)
      CALL USWFV('GQGT',4,GQGT,N, 1,1)
C
      DO 3 I=1,N
    3 READ(5,99) (AK(I,J),J=1,3)
      CALL USWFM('K',1,AK,8,N,3,1)
C
      DO 4 I=1,3
    4 READ(5,99) (H(I,J),J=1,N)
      CALL USWFM('H',1,H,3,3,N,1)
C
      DO 5 I=1,3
    5 READ(5,99) (R(I,J),J=1,3)
      CALL USWFM('R',1,R,3,3,3,1)
C
      DO 6 I=1,136
    6 VAR(I)=0.
      DO 7 I=1,N
      DO 7 J=1,N
C
    7 DF(I,J)=FS(I,J)-F(I,J)
      CALL USWFM('DEL F',5,DF,8,N,N,1)
      CALL VMULFF(AK,R,N,3,3,8,3,TMP1,8,IER)
      CALL VMULFP(TMP1,AK,N,3,N,8,8,AKRKT,8,IER)
      CALL USWFM('KRKT',4,AKRKT,8,N,N,1)
C
C     CALCULATE THE DIFFERENCE BETWEEN THE DYNAMICS
C     IMPLEMENTED IN THE FILTER AND THE PLANT, DF=FS-F
C
      DO 20 I=1,N
      DO 20 J=1,N
      DFT(I,J)=DF(I,J)
   20 FT(I,J)=F(I,J)
      CALL VTRANX(DFT,N,N,8)
      CALL USWFM('DEL FT',6,DFT,8,N,N,1)
      CALL VTRANX(FT,N,N,8)
      CALL USWFM('FT',2,FT,8,N,N, 1)
      CALL VMULFF(AK,H,N,3,N,8,3,TMP1,8,IER)
C
      DO 21 I=1,8
      DO 21 J=1,8
      FSMKH(I,J)=FS(I,J)-TMP1(I,J)
   21 FSMKHT(I,J)=FSMKH(I,J)
      CALL USWFM('FS-KH',5,FSMKH,8,N,N,1)
      CALL VTRANX(FSMKHT,N,N,8)
      CALL USWFM('(FS-KH)T',8,FSMKHT,8,N,N, 1)
      T=0.
      TOL=1.D-5
      IND=1
      L=0
      IF(N.EQ.8) GO TO 11
C
      DO 31 I=1,NS
      L=L+1
   31 VAR(L)=U(I)
C
      DO 32 I=1,N
      DO 32 J=1,N
      L=L+1
   32 VAR(L)=V(I,J)
C
      DO 33 I=1,N
      L=L+1
   33 VAR(L)=P(I)
   11 DO 10 K=1,NP
```

```
C      TEND=FINAL TIME
C
       TEND=K*DT
C
C      DVERK SUBROUTINE FINDS THE SOLUTION OF THE SYSTEM OF
C      DIFFERENTIAL EQUATIONS
C
       CALL DVERK(NV,FUN,T,VAR,TEND,TOL,IND,C,136,W,IER)
       IF(IND.LE.0.OR.IER.NE.0) STOP
       CALL VCVTSF(VAR(N1),N,PFULL,8)
C
C      CALCULATE AND PRINT THE RMS ESTIMATE ERRORS
C
       DO 30 I=1,N
       REAP=PFULL(I,I)
       PFULL(I,I)=DABS(REAP)
   30  PSQR(I)=DSQRT(PFULL(I,I))
       WRITE(6,90) T,(PSQR(I),I=1,N)
   90  FORMAT('0T=',F10.5,' PSR=',8G14.7)
C
C      IF DESIRED PRINT THE  COVARIANCE MATRICES,P,U AND V
C      CALL USWSM('U',1,U,N,3)
C      CALL USWFM('V',1,VAR(NS+1),8,N,N,3)
C      CALL USWSM('P',1,VAR(N1),N,3)
   10  CONTINUE
       STOP
       END
       SUBROUTINE VTRANX(A,N,NC,IA)
       IMPLICIT REAL*8 (A-H,O-Z)
       DIMENSION A(IA,IA),B(8,8)
C
       DO 1 I=1,N
       DO 1 J=1,N
    1  B(I,J)=A(J,I)
C
       DO 2 I=1,N
       DO 2 J=1,N
    2  A(I,J)=B(I,J)
       RETURN
       END
       SUBROUTINE FUN(NV,T,VAR,DRV)
C
C      FCN SUBROUTINE IS USED FOR EVALUATING FUNCTIONS(INPUT)
C
C
       IMPLICIT REAL*8 (A-H,O-Z)
       COMMON F(8,8),FS(8,8),GQGT(8),AK(8,3),R(3,3),
      *AKRKT(8,8),DP(8,8),FT(8,8),PSHKHT(8,8),DPT(8,8),
      *PSHKH(8,8),H(3,8)
       COMMON/KTR/N,NS,NPD
       DIMENSION U(36),V(8,8),P(35),UD(36),VD(8,8),
      *VAR(136),DRV(136),C(24),WK(136,9),PD(36)
       DIMENSION TMP1(8,8),TMP2(8,8),TMP3(8,8)
C
       L=0
       DO 1 I=1,NS
       L=L+1
    1  U(I)=VAR(L)
C
       DO 2 I=1,N
       DO 2 J=1,N
       L=L+1
    2  V(I,J)=VAR(L)
C
       DO 3 I=1,NS
       L=L+1
    3  P(I)=VAR(L)
       IF(T.EQ.0) KT=NPD
       KT=KT+1
       IF (KT.GE.5) GO TO 15
       WRITE(6,99) T
   99  FORMAT('0T=',D25.15)
```

```
      CALL USWSM('U',1,U,N,3)
      CALL USWFM('V',1,V,7,N,N,3)
      CALL USWSM('P',1,V,N,3)
   15 CALL VMULFS(P,U,N,N,8,TMP1,8)
      IF(KT.LT.5) CALL USWFM('PU',3,TMP1,8,N,N,3)
      CALL VMULSP(U,N,PT,N,8,TMP2,8)
      IF(KT.LT.5) CALL USWFM('UPT',3,TMP2,8,N,N,3)
C
      DO 5 I=1,N
      DO 4 J=1,N
    4 TMP1(I,J)=TMP1(I,J)+TMP2(I,J)
    5 TMP1(I,I)=TMP1(I,I)+GOGT(I)
      CALL VCVTFS(TMP1,N,7,UD)
      IF(KT.LT.5) CALL USWSM('UDOT',4,UD,N,3)
      CALL VMULFF(P,V,N,N,N,8,8,TMP1,8,IER)
      IF(KT.LT.5) CALL USWFM('PV',3,TMP1,8,N,N,3)
      CALL VMULFF(V,FSMKHT,N,N,N,8,8,TMP2,8,IER)
      IF(KT.LT.5) CALL USWFM('V(FS-KH)T',10,TMP2,8,N,N,3)
      CALL VMULSP(U,N,DPT,N,8,TMP3,8)
      IF(KT.LT.5) CALL USWFM('U*DPT',5,TMP3,8,N,N,3)
C
      DO 7 I=1,N
      DO 6 J=1,N
    6 VD(I,J)=TMP1(I,J)+TMP2(I,J)+TMP3(I,J)
    7 VD(I,I)=VD(I,I)-GOGT(I)
      IF(KT.LT.5) CALL USWFM('VDOT',4,VD,8,N,N,3)
       CALL VMULFS(FSMKH,P,N,N,8,TMP1,8)
      IF(KT.LT.5) CALL USWFM('(FS-KH)P',8,TMP1,8,N,N,3)
      CALL VMULSP(P,N,FSMKHT,N,8,TMP2,8)
      IF(KT.LT.5) CALL USWFM('P(FS-KH)T',9,TMP2,8,N,N,3)
      CALL VMULFF(DP,V,N,N,N,8,8,TMP3,8,IER)
      IF(KT.LT.5) CALL USWFM('DP*V',4,TMP3,8,N,N,3)
C
      DO 8 I=1,N
      DO 8 J=1,N
    8 TMP1(I,J)=TMP1(I,J)+TMP2(I,J)+TMP3(I,J)
      CALL VMULFM(V,DP,N,N,N,8,8,TMP3,8,IER)
      IF(KT.LT.5) CALL USWFM('VT*DP',5,TMP3,8,N,N,3)
C
      DO 10 I=1,N
      DO 9 J=1,N
    9 TMP3(I,J)=TMP1(I,J)+TMP3(I,J)+AKRKT(I,J)
   10 TMP3(I,I)=TMP3(I,I)+GOGT(I)
      IF(KT.LT.5) CALL USWFM('PDOT',4,TMP3,8,N,N,3)
      CALL VCVTFS(TMP3,N,8,PD)
      IF(KT.LT.5) CALL USWFV('PDOT(SYM)',9,PD,N,1,3)
      L=0
C
      DO 11 I=1,NS
      L=L+1
   11 DRV(L)=UD(I)
C
      DO 12 I=1,N
      DO 12 J=1,N
      L=L+1
   12 DRV(L)=VD(I,J)
C
      DO 13 I=1,NS
      L=L+1
   13 DRV(L)=PD(I)
      IF (KT.LT.5) CALL USWFV('DRV',3,DRV,NV,1,3)
      RETURN
      END
```

## LIST OF REFERENCES


1. Agard, L. S. 95, Strapdown Inertial Navigation Systems, NATO Publication, France, 1978.

2. Maybeck, P. S., Stochastic Models, Estimation and Control, Vol. 1, Academic Press, 1979.

3. Bryson, A. E., Kalman Filter Divergence and Aircraft Motion Estimators, Vol. 1, AIAA Journal, January 1978.

4. Matallana, J. A., Sensitivity of The S.K.F. to Stability Derivatives Variations in An I.N.S., Master's Thesis, Naval Postgraduate School, Monterey, 1980.

5. Gelb, A. and others, Applied Optimal Estimation, MIT Press, 1974.

6. Franklin, G. F. and Powell, J. D., Digital Control of Dynamic Systems, Addison-Wesley, 1980.

7. Anderson, B. D. O., and Moore, J. B., Linear System Optimization with Prescribed Degree of Stability, Proc. IEE, Vol. 116, No. 12, December 1969.

8. Collins, D. J., Missiles, Navigation and Avionic Systems, Class Notes, Naval Postgraduate School, October 1981.

9. Walker, R., OPTSYS 4 at SCIP Computer Program, Stanford University, Aero/Astro Department, December 1979.

10. Potter, G. G., An Aid to Using OPTSYS at NPS, Class Term Paper, Naval Postgraduate School, March 1982.

11. Stansell, T. A., Jr., The Many Faces of Transit, Vol. 25, No. 1, Journal of the Institute of Navigation, Spring 1978.

## INITIAL DISTRIBUTION LIST

| | | No. Copies |
|---|---|---|
| 1. | Defense Technical Information Center<br>Cameron Station<br>Alexandria, Virginia 22314 | 2 |
| 2. | Library, Code 0142<br>Naval Postgraduate School<br>Monterey, California 93940 | 2 |
| 3. | Department Chairman, Code 62<br>Department of Electrical Engineering<br>Naval Postgraduate School<br>Monterey, California 93940 | 1 |
| 4. | Professor D. J. Collins, Code 67Co<br>Department of Aeronautics<br>Naval Postgraduate School<br>Monterey, California 93940 | 2 |
| 5. | Professor H. A. Titus, Code 62Ts<br>Department of Electrical Engineering<br>Naval Postgraduate School<br>Monterey, California 93940 | 1 |
| 6. | Lcdr. Gary G. Potter<br>Charles Stark Draper Laboratory<br>555 Tecnnology Square<br>Cambridge, Massachusetts 02139 | 2 |